Gangi, Nicolás

Implementación de redes mesh para IoT

Tesis para la obtención del título de grado de Ingeniero Electrónico

Director: Marcucci, Ricardo Martín

UNIVERSIDAD
CATOLICA DE CORDOBA

Universidad **Jesuita**

TRABAJO FINAL

Título: Implementación de redes mesh para IoT

Autor: Nicolás Gangi

Tutor:

11 October 2019

ACEPTACION DEL TRABAJO FINAL

Universidad Católica de Córdoba
Facultad de Ingeniería
Carrera de Ingeniería electrónica

Título: Ingeniero electrónico

Autor: Nicolás Gangi

Calificación:

……..

Firma y Aclaración de Presidente de Mesa Examinadora

……..

Firma y Aclaración de Vocal de Mesa Examinadora

……..

Firma y Aclaración de Vocal de Mesa Examinadora

Córdoba, …… de ……. De 2020

Prólogo

En el presente trabajo final de grado titulado *Implementación de redes mesh para IoT* se estudia y analiza las ventajas que conlleva el uso de la topología de **red mesh** para el **internet de las cosas**. A lo largo del informe se podrá observar el desarrollo que se le dio a este proyecto y como el mismo se fue llevando a cabo, partiendo desde una introducción teórica al internet de las cosas con su definición, ideas, virtudes y limitaciones.

Una vez obtenida y entendidas las limitaciones se puede comenzar a proponer las soluciones que puedan mejorar esta condición. En este caso se abordará la temática de topologías de red, en especial las **mesh** de las cuales se hará un estudio ya análisis para entenderlas mejor. Es en este momento cuando se redacta sobre la incorporación de **redes mesh** para el internet de las cosas y cómo esta topología puede traer ventajas para solucionar limitaciones que acarrea esta tecnología.

Por otro lado se detalla los pasos a seguir en el proceso de prototipo de placas electrónicas **PCB** y como partiendo de una idea de circuito se puede llegar a obtener algo funcional y que a su vez encaje con el objetivo que se busca. Como todo en la vida, esto tiene sus complicaciones y limitaciones las cuales también se detallan cómo por ejemplo los problemas que se encuentran al momento de comprar los componentes y soldarlos a la placa. Por último y se podría decir que la parte más divertida es la de carga el *software* diseñado previamente al *hardware* para comprar su uso en conjunto y ver si se lograron satisfacer las limitaciones y problemas que se habían encontrado en un principio.

Esta tesis contiene gráficos y fotografías que ayudan al lector a ubicarse y poder entender de una manera más sencilla los pasos que se llevaron a cabo y las metodologías utilizadas. Se trata de un trabajo que busca englobar la mayor cantidad de contenidos vistos a lo largo de la carrera ingeniería en electrónica y ser utilizados para lograr un producto funcional y con un gran impacto social.

Resumen

En un mundo en el cual queremos tener todo conectado y poder manejar cada centímetro de nuestra casa desde nuestro teléfono móvil cada solución innovadora que se cruza frente a nuestros ojos nos llama la atención. Desde saber la temperatura de nuestra hogar cuando estamos de viaje o prender una luz desde la otra punta del mundo, hasta poder encender el aire acondicionado en nuestro camino a casa.

Dado esta demanda y este mercado en crecimiento es que se vio la necesidad del desarrollo de módulos que cumplan y satisfagan estas nuevas demandas creadas por la propia globalización pero mejorando algunos aspectos débiles con los que constan los dispositivos actuales por medio de la implementación de una topología de **red mesh**. Por lo que se partirá del estudio de mercado que se tuvo que hacer, pasando por diseño, fabricación, procesos de prueba y puesta en marcha hasta lo que es hoy en día, un sistema mallado con una conexión robusta y que sin importar la cantidad de nodos se mantenga en funcionamiento.

Abstract

In a world in which we want to have everything connected and be able to manage every inch of our house from our cell phone, every innovative solution that crosses in front of our eyes catches our attention. From knowing the temperature of our home when we are travelling or turning on a light from the other side of the world, to being able to turn on the air conditioning on our way home.

Given this demand and this growing market is that we saw the need to develop modules that meet and satisfy these new demands created by globalization itself but improving some weak aspects that current devices have through the implementation of a **mesh network** topology. Therefore, we will start from the market study that had to be done, going through design, manufacturing, testing and implementation processes until what it is today, a mesh system with a robust connection and that no matter the number of nodes is kept in operation.

Objetivos

GLOBALES

- Lograr el desarrollo de una topología de red Mesh.
- Desarrollar un producto para **IoT** el cual permita conectar las cosas del entorno a internet para su respectivo control.
- Llevar al mercado un producto innovador que utilice tecnología actual, fácil de controlar y que su uso traiga grandes beneficios.
- Resolver las limitaciones que tiene al día de hoy los dispositivos **IoT**.

ESPECÍFICOS

- Realizar un proyecto como trabajo final para obtener el título de ingeniero en electrónica.
- Desarrollar un proyecto completo que englobe tanto hardware y software, siendo este último el de mayor dimensión.
- Lograr experiencia con la fabricación de placas electrónicas en el territorio del gran país asiático de China debido al bajo valor de producción que se consigue.
- Lograr entender e interiorizar el proceso de desarrollo de software, desde la etapa de requerimientos hasta la puesta en práctica del firmware final.
- Aprender a usar las tecnologías actuales tanto para hardware como para software.
- Ampliar el horizonte de conocimiento de lenguajes de programación.
- Conocer nuevas herramientas que se pueden utilizar en un futuro para la administración y gestión de proyectos.
- Trabajar respetando las condiciones y tiempos de entrega predeterminados, similar a una experiencia profesional completa y real.

Marco teórico

INTRODUCCIÓN

Presentación del tema

Nos encontramos en el siglo 21, la época de la conectividad, una era en la que todo se conecta con todo y no solo eso, sino que lo hace de una manera muy fácil. Podemos entablar una conversación en tiempo real con una persona que se encuentra al otro lado del mundo y sentir que estamos uno al lado del otro, podemos prender nuestro televisor incluso sin estar en casa y millones de otros usos que seguiríamos nombrando. Pero todo esto no sería posible sin el tan famoso **internet**. Si buscamos su definición nos dice que "la internet es un sistema de red conectado globalmente que se utiliza para trasmitir datos a través de varios tipos de medios. La internet es una red de intercambios globales - incluyendo redes privadas, publicas, empresariales, académicas y gubernamentales - conectadas por tecnologías guiadas, inalámbricas y de fibra óptica."

Como se dijo anteriormente, una de las formas de conectarse a internet es inalámbricamente, de eso se basa este proyecto final. Pero, ¿Cuál es el fin de la intercomunicación e interconexión? Hay una palabra muy famosa que corre por estos días y es **IoT**, del ingles ***Internet of Things***, o también conocida como **Internet de las cosas**. El internet de las cosas es un concepto que se refiere a la interconexión digital de objetos cotidianos con internet, en definitiva, es la conexión de internet más con objetos que con personas. Es como si cada instrumento de la vida cotidiana tuviese una etiqueta que nos permitiera controlarlo desde nuestro teléfono móvil, algo increíble y que todo el mundo quisiera tener.

El internet de las cosas constituye un cambio radical en la calidad de vida de las personas de la sociedad, ofrece una gran cantidad de nuevas oportunidades de acceso a datos, servicios específicos en la educación, seguridad, asistencia sanitaria y en el trasporte, entre otras tantas cosas. Este concepto fue propuesto en el año 1999 por Kevin Ashton en el MIT, donde se realizabas investigaciones en el campo de la identificación por radiofrecuencia en red y tecnologías de sensores.

Con respecto a los objetos los cuales pueden formar parte de esta interconexión podrían ser cualquiera, desde sensores y dispositivos mecánicos hasta objetos cotidianos como pueden ser una heladera, una lavadora o simplemente ropa. Cualquier cosa que se pueda imaginar podría ser conectada a internet e interactuar sin necesidad de la intervención

humana, el objetivo por tanto es una interacción de maquina a maquina, o lo que se conoce como una interacción **M2M**, del ingles ***machine to machine***, o dispositivos M2M.

¿Por qué esta de moda el IoT?¿Qué aplicaciones tiene?

Internet ha evolucionado rápidamente y esto ah permitido que **IoT** sea ya una realidad y no solo una visión de futuro. La fama de esta tecnología radica principalmente en todas las aplicaciones y posibilidades que nos proporciona tanto para mejorar la vida cotidiana de las personas como los entornos empresariales, donde ya se esta implantando desde hace ya algún tiempo. Las aplicaciones son casi infinitas, pero se van a describir algunos ejemplos para dar visibilidad de alguna de ellas, tanto en la vida cotidiana como en el entorno empresarial:

- Supongamos la heladera de una casa, donde se conservan alimentos que, a su vez, tienen una fecha de caducidad. En este escenario, se podría conectar la heladera a internet para que le avisara al usuario a través de su teléfono móvil, por ejemplo, de cuando caducan los alimentos, si hay una baja de temperatura por alguna avería, si algún alimento se está acabando o simplemente el consumo eléctrico en base al numero de veces que se abre la puerta de la misma.

- Otro escenario podría ser el de la domótica, donde ya hay numerosos dispositivos que se conectan a internet para facilitar la vida de los seres humanos, véase por ejemplo los dispositivos controlados por voz a los que se les solicita que reproduzcan una canción desde un repositorio de internet, o los dispositivos y aplicaciones que permiten controlar todos los parámetros del agua de un acuario, o incluso los sistemas de alarmas de las casa que se conectan con las centrales. Los sistemas de seguridad que se conectan a la red para avisarte cuando alguien entra en tu casa o aquellos dispositivos que permiten encender la calefacción desde un teléfono móvil.

- Si se piensa en aplicaciones industriales, **IoT** es usado ya en muchas plantas de producción donde los dispositivos y sensores conectados a la red permiten analizar los datos y generar alarmas y mensajes que son enviados a los dispositivos usuarios para que tomen las acciones necesarias o incluso iniciar protocolos de actuación de forma automática, sin interacción humana, para corregir o tratar dichas alarmas.

- Otro ejemplo de aplicación seria el sector ganadero donde la monitorización biométrica y la geolocalización es un factor que ayuda a los ganaderos a que sus animales estén siempre controlados.

Términos muy relacionados con **IoT** pueden ser "*Smart Cities*", "*Smart Buildings*" donde se utilizan dispositivos **IoT** para mejorar el control del tráfico, el control de los suministros de agua y calefacción en un edificio, el control del transporte público, etc.

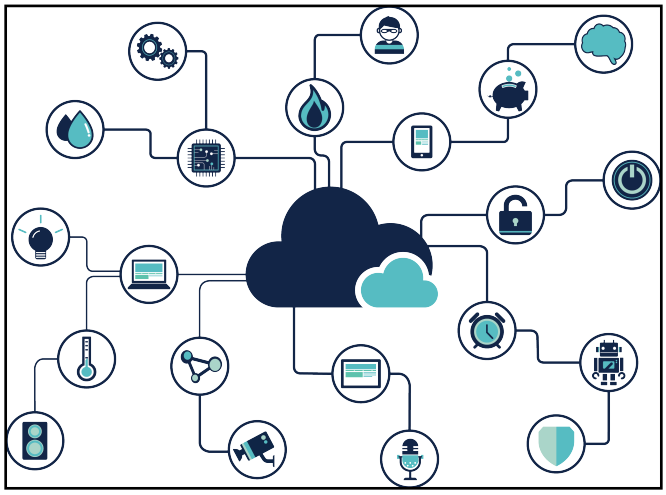


Imagen Nº1: IoT idea

¿Qué tecnologias se utilizan?

Este proceso de comunicación es donde **IoT** esta evolucionando ya que uno de los problemas a salvar es del tipo de protocolo con el que se comunican dichos dispositivos (es decir, "el idioma" que hablan entre ellos). Actualmente existen dispositivos o sensores muy nuevos cuya comunicación y conexión a internet es fácil y directa, pero también existen muchos otros dispositivos mas antiguos no estándar cuyo protocolo de comunicación y conexión no es trivial, es ahi donde viene uno de estos problemas a salvar. Adicionalmente, cada fabricante tiene sus propios protocolos de comunicación que hace que no todos los dispositivos sean compatibles. Uno de los mecanismos que se ha intentado establecer es la creación de un protocolo abierto y y estándar (propuesto por **IBM**) denominado **MQTT** (*Message Queuing Telemetry Transport*), que permite que todos los fabricante puedan participar y soportarlo, facilitando así la comunicación entre distintos dispositivos de diferentes fabricantes.

Otra parte importante de estos dispositivos son los sensores; el procesador y la plataforma se encarga de gestionar la información, pero esta, debe provenir de los sensores. En este sentido, Arduino ha permitido que este tipo de tecnologías este al alcance de todos los usuarios y es por esto que este trabajo a su vez cuenta con sensores Arduino.

Finalmente, otro componente importante tecnológicamente para habilitar el **IoT** es la tecnología utilizada para la comunicación entre varios dispositivos cuya ubicación no sea próxima, es decir, las redes de comunicación. En este apartado se puede hablar por ejemplo de comunicaciones a través de una red "**WiFi**" que, aunque admite una tasa de transferencia alta, tiene un consumo alto y bajo alcance. Otro ejemplo conocido seria una red móvil (**3G**, **4G** o la futura **5G**) donde el alcance sería mayor y de menor consumo. Adicionalmente, existen otro tipo de redes específicas para **IoT** cómo puede ser **Sigfox** (con gran cobertura tanto en Estados Unidos como en Europa) o **LoRa**.

Topologías

La topología de red se define como el mapa físico o lógico de una red para intercambiar datos, en otras palabras, es la forma en que está diseñada la red, sea en el plano físico o lógico. El concepto de red puede definirse como "conjunto de nodos interconectados". En el caso de este proyecto estamos hablando de una red WiFi.

Un ejemplo claro de esto es la topología de árbol, la cual es llamada así por su apariencia estética (**Imagen Nº1**), por la que se puede comenzar con la inserción del servicio de internet desde el proveedor, pasando por el *router*, luego por un *switch* y este deriva a otro *switch* u otro *router* o sencillamente a los *hosts*, el resultado de esto es una red con apariencia de árbol porque desde el primer *router* que se tiene se ramifica la distribución de internet, dando lugar a la creación de nuevas redes o subredes tanto internas como externas. Además de la topología estrella, se puede dar una topología lógica a la red y eso dependerá de lo que se necesite en el momento.

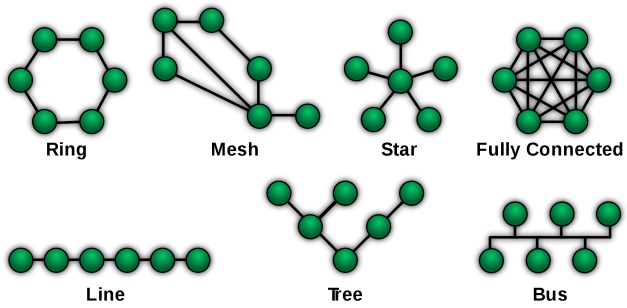


Imagen Nº2: Topologías de redes

Los componentes fundamentales de una red son el servidor, los terminales, los dispositivos de red y el medio de comunicación. En algunos casos, se puede usar la palabra arquitectura en un sentido relajado para hablar a la vez de la disposición física del cableado y de cómo el protocolo considera dicho cableado. La topología de red la determina únicamente la configuración de las conexiones entre nodos. La distancia entre los nodos, las interconexiones físicas, las tasas de transmisión y los tipos de señales no pertenecen a la topología de la red, aunque pueden verse afectados por la misma.

Algunas topologías básicas son:

- Punto a punto (*point to point*, PtP).
- En bus (conductor común) o lineal.
- En estrella (*start*).
- En malla (*mesh*).
- En árbol (*tree*) o jerárquica.
- Topología híbrida, combinada o mixta.
- Cadena margarita (*daisy chain*).

Mesh networking

Una **Mesh Network** (Red de malla) o **Meshnet** es una topología de red local en la que los nodos de la estructura (es decir, puentes, *Switches* y otros dispositivos de infraestructura) se conectan de forma directa, dinámica y no jerárquica a tantos otros nodos como sea posible y cooperan entre si para encaminar eficazmente los datos desde y hacia los clientes. Esta falta de dependencia de un nodo permite que cada nodo participe en la transmisión de la información. Las **redes Mesh** se autoorganizan y autoconfiguran dinámicamente, lo que puede reducir los gastos de instalación. La capacidad de autoconfiguración permite una distribución dinámica de las cargas de trabajo, en particular en el caso de que unos pocos nodos fallen. Esto a su vez contribuye a la tolerancia a las fallas y a la reducción de los costos de mantenimiento.

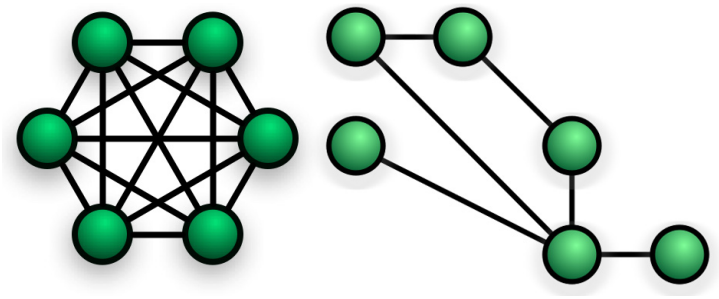


Imagen Nº3: Mesh network

A priori puede parecer que hacen lo mismo que un router actual con varios repetidores, pero no es así. Para empezar, los repetidores no se comunican todos ellos entre sí sino que lo hacen habitualmente solo con los router. En el caso de las redes mesh, su punto fuerte está precisamente en la gestión avanzada de los elementos de la red. Los sistemas mesh no nos conectan al punto más cercano sino al que, aunque esté más alejado del dispositivo, nos dará la mejor señal atendiendo a múltiples variables de la red.

Una red de malla es capaz de dirigir el trafico por la red siempre de la forma más optima para disponer de la mejor señal posible, calculan a que nodo es mejor conectarse en cada momento según el estado de otros nodos, los dispositivos conectados, la distancia a cada uno de los nodos, potencia de la señal y otros muchos factores, de forma completamente transparente al usuario, el cual no tiene que preocuparse de a que nodo está conectado.

Esa gestión inteligente del tráfico y situación de la red es la otra gran diferencia con los repetidores, a los cuales los dispositivos se suelen conectar según su proximidad y no la situación real de la red. Así, aunque un nodo cayera, en una red mesh no perderíamos la señal porque el sistema automáticamente derivaría el tráfico por otros nodos de la red, los cuales se pueden conectar a cualquier otro nodo de la red mallada.

En el caso de este proyecto, a diferencia de la mayoría de los usos que se le da a una **meshnet** que es como red WiFi de internet, si se usará la tecnología WiFi pero no para transmitir datos de internet sino como medio de comunicación entre nodos.

Normalmente se podría llegar a confundir la topología estrella con **mesh** pero dado que son dos formas de en las que los módulos se conectan y tienen sus diferencias, es una topología con la que se puede comparar. La tradicional topología en estrella esta construida en una forma que todos los nodos se conectan con el host central, esto genera una estructura muy dependiente y propensa a fallas en la conexión. En el caso de que un nodo deje de funcionar, no hay manera de que se envíen o reciban datos de el.

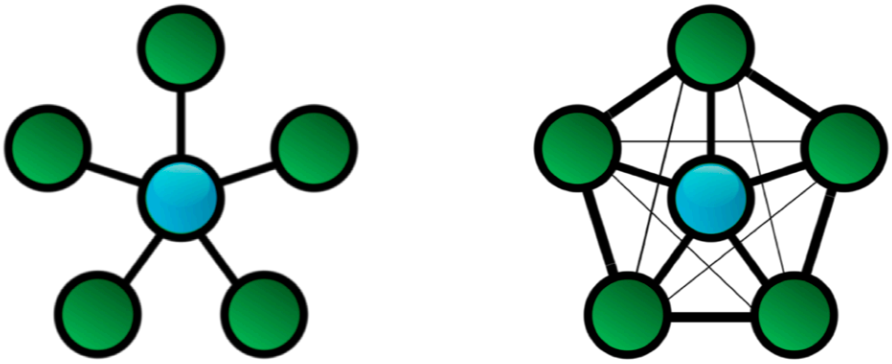


Imagen Nº4: Comparación red estrella vs red mesh

¿Cómo funciona una red mesh?

La cantidad de componentes que forman parte de una red mesh puede variar pero usualmente rondan entre los 2 y 4 elementos.

- *Nodos:* Toda red mesh tiene nodos, los cuales son los dispositivos que se comunican entre ellos. Es así como la información se envía y recibe.

- *Host central:* También una **red mesh** consta de un nodo central que no solo se comunica con los nodos sino que también lo hace con internet (puede tener intermediarios como por ejemplo una **Raspberry**). Es la manera en la que la información interna de la red sale al exterior y también la forma en que la información de exterior entra a la red.

- *Repetidores:* Dispositivos que mantienen un determinado nivel de señal. En el caso de este trabajo no son usados los repetidores.

- *End-point:* En algunos casos las redes mesh operan de formas en las que algunos nodos no reciben información, solamente la envían hacia otros nodos, esto es por lo que son llamados puntos finales (*end-points*). Nuevamente, este proyecto no incluye nodos de este tipo por lo que no se profundizara más en ellos.

Las redes mesh para dispositivos **IoT** pueden transferir mensajes de 2 formas: por *flooding* o por *routing. Flooding* es una técnica donde todos los todos actúan como repetidores de información. Esto permite que el tiempo desde que el mensaje es enviado hasta que es recibido es muy corto. Por consecuencia, esta técnica consume mucha energía debido que un mismo mensaje es enviado por todos los caminos posibles pero solo el más corto es el que termina entregando la información. Es por esto que el *flooding* no es una técnica recomendable en los casos que el consumo energético es un factor de gran importancia.

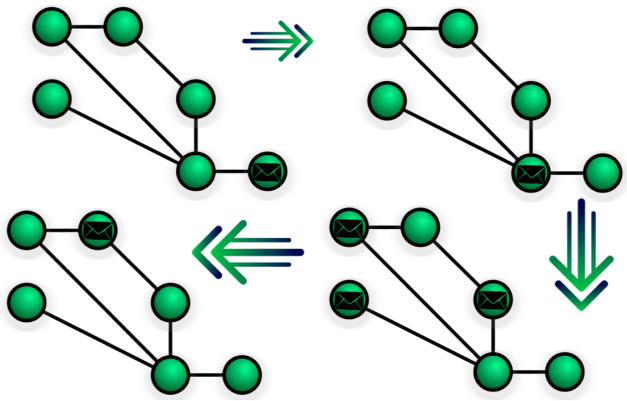


Imagen Nº5: Técnica *flooding*

Por otro lado, *routing* elige solamente un camino desde un nodo hacia el otro para transferir el paquete hasta el destinatario. En comparación a *flooding, routing* envía el mensaje un nodo a la vez, solamente cuando la conexión es imposible este mismo elige otra ruta. Para asegurarse que el camino por el que se optó es el más eficiente, la red usa el algoritmo de “enviar por el camino más corto” (*Shortest Path Bridging*) o también llamado SPB o algoritmo de IEEE 802.1aq. Este algoritmo nos permite que la información sea transferida por el camino disponible más corto.

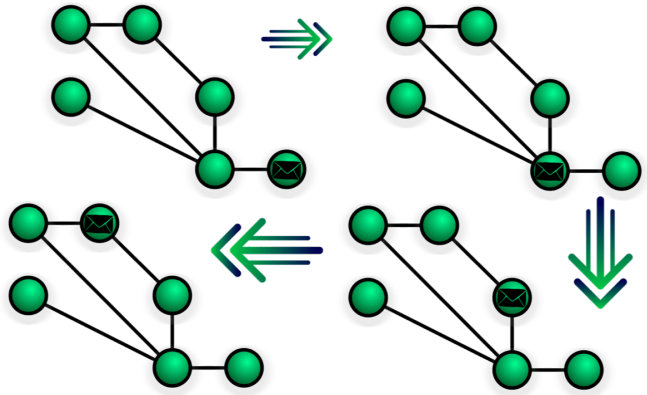


Imagen Nº6: Técnica *routing*

Full mesh network vs. Partial mesh network

Dentro de la topología mesh nos encontramos a su vez otras sub topologías y, a su vez, cada una con sus ventajas y desventajas. Existen dos tipos posibles de **topologías mesh**, la mesh completa (*full mesh network*) y la topología de mesh parcial (p*artial mesh network*).

En la **topología de mesh completa** se requiere que cada uno de los dispositivos que la conforman, ya sea nodos o host, se encuentren conectados entre si formando una mallado, o más fácilmente explicado cada nodo debe conectarse directamente a todo el resto de los nodos. Esto nos proporciona una completa redundancia y un máximo rendimiento debido a que un paquete de información puede ser enviado y recibido por un solo enlace. Esta manera de conexionado está destinada a redes de tamaño reducido debido a que se debe generar un vinculo entre cada uno de los nodos con los demás.

Por otro lado, se puede conectar a los dispositivos usando la topología de **red mesh parcial**. En este caso los dispositivos no se conectan entre si directamente sino que cada uno se encuentra conectado al menos a otros dos nodos generando así varias rutas por las cuales pueden enviarse paquetes. Esto nos permite que la información que se envíe vaya saltando de un nodo a otro. Los tiempos de envío de paquete en el caso de la **red mesh parcial** son mayores si lo comparamos con la topología completa debido a que la mayoría de las veces el destinatario no se encontrará cómo nodo vecino. Contrario a esto, la conexión parcial nos permite extender la red obteniendo así mayor alcance gracias a que el conexionado de los nodos no tiene que ser tan extenso. Cabe destacar la simplicidad de la topología parcial comparado a la completa.

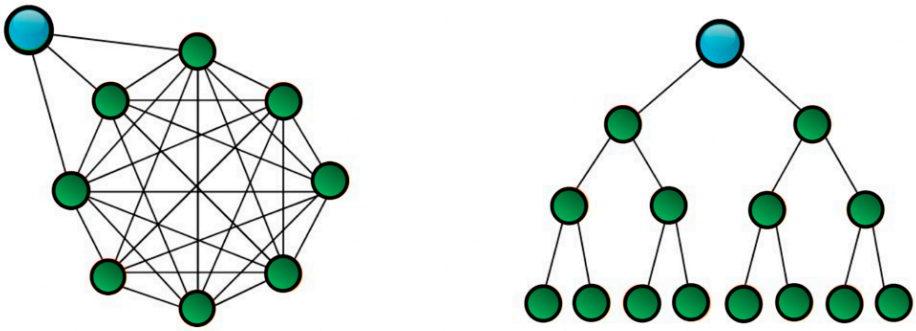


Imagen Nº7: Full mesh vs. Partial mesh

Beneficios de las redes mesh

Las **redes mesh** tienen algunos beneficios muy significativos que las convierten en una topología muy conveniente a la hora de diseñar una red. Para decepción de algunos y suerte de otros, no son muy utilizadas debido a su falta de populatidad.

- Auto-curación (*Self-healing*): El algoritmo de "auto-curación" también mencionado anteriormente como "*Shortest Path Bridging*" o "enviar por el camino más corto" elige automáticamente la mejor ruta para enviar los datos, incluso si ciertos nodos han perdido la conexión. El algoritmo utiliza solo las conexiones que están disponibles y funcionando para así poder calcular la mejor ruta. Gracias a ellos es que, aunque habiendo algunos dispositivos que dejen de funcionar, toda la red sigue enviando y recibiendo la información necesaria para mantener la tarea que se está llevando a cabo.

- Auto-configuración (*Self-configuring*): Gracias al autodescubrimiento las **redes mesh** son capaces de auto-configurarse, esto significa que los nuevos nodos se calibran automáticamente y se conectan a la red sin ninguna configuración previa. Esto hace que la red sea muy fácil de administrar y expandir.

- Escalabilidad y fiabilidad (*Scalability and Reliability*): Lo bueno de esta topología de red es que es extremadamente fácil de escalar, los nodos pueden ser añadidos o quitados sin ningún problema de eficiencia. Normalmente, cuanto más dispositivos tenga la red, mayor es el problema. Sin embargo, pasa todo lo contrario cuando se trata de una **red mesh**, cuantos más nodos haya más rutas habrá. La **red mesh** funciona de manera que la información se envía por la ruta más rápida. Añadir nodos crea más formas en las que los paquetes pueden viajar. Esto hace que la red sea más rápida y resistente a errores.

- Reducción de costos (*Cost reduction*): Debido a que la **red mesh** no necesita conexión a internet, utiliza muy poca energía. Los sensores también son muy baratos y duran muchos años. Esta topología de red es rentable de dos maneras, la primera, no necesita conexión a internet como se menciono recién por lo que usa muy poca energía.

Desventajas de las redes mesh

Aunque hay muchos beneficios en las **redes mesh**, también existen algunas desventajas. Por eso es tan importante tener un conocimiento completo y profundo antes de decidir si la topología mesh es la mas adecuada para lo que se desea hacer.

- Baja capacidad (*Low capacity*): La **red mesh** es la más efectiva cuando se usa para enviar pequeños paquetes de información. Desafortunadamente, no funciona bien en condiciones en que se necesitan mandar paquetes de tamaño mayor como video. En situaciones en las que el envío de grandes informaciones es obligatorio, la mejor opción sería usar la **red mesh WiFi**.

- Latencia (*Latency*): Saltar de un nodo a otro puede ralentizar el proceso de recepción y envío de datos, no es un problema cuando el sistema necesita un paquete solo cada pocos minutos o menos. Sin embargo, puede que no sea suficiente para algunos otros sistemas en donde el tiempo necesario sea mucho menor. Una **topología mesh completa** puede acelerar el proceso conectando cada nodo entre sí.

- Mantenimiento (*Maintenance*): Debido a que las redes tienen una capacidad de auto-curación, encontrar un nodo que no funcione puede llevar bastante tiempo. La red se crea de forma que funcione correctamente aunque no todos los nodos estén disponibles. Esto significa que no sabremos cuándo el nodo tiene un problema. Por otro lado, las **redes mesh** se implementan para hacer el sistema más inteligente y eficiente. Por lo general, esto también significa recibir información sobre el rendimiento de cada nodo individualmente.

Redes Mesh en redes IoT

Si hacemos memoria recordaremos que el **internet de las cosas** es uno de los temas mayormente hablados tanto en la industria como en el mundo de las tecnologías de información y comunicación. Por otro lado, las **redes Mesh**, una topología de red que ha sido discutida por décadas y que no fue puesta en uso a gran escala al día de hoy. Ambas tecnologías son muy potentes por si mismas pero el hacerlas trabajar juntas puede significar una gran diferencia en la nueva era.

Con el importante crecimiento de la industria de los semiconductores, la creación de pequeños dispositivos con una poderosa capacidad de procesamiento y conectividad ya no es un sueño para los ingenieros como lo era antes. Actualmente, la mayor parte de la investigación de los dispositivos habilitados para **IoT** es la de recolección y procesamiento de datos, es decir, la creación de nuevos sensores. Sin embargo, la red que integran este tipo de dispositivos a la internet se suele dejar intacta con el simple uso de las soluciones de redes informáticas existentes, como **WLAN** o **Bluetooth**. Estas redes informáticas no están diseñadas para dispositivos de baja potencia como los sensores remotos, incluso podríamos considerar a estos dispositivos como mini computadoras. La naturaleza al fallo de la conexión de punto único de estas redes hace que todo el sistema sea extremadamente vulnerable cuando se trata de desastres o incluso de un entorno difícil, ya que los sensores pueden tener que ser desplegados en algunos lugares de dificil acceso. Además, la capacidad del *hub* central de la red también puede limitar la cobertura del servicio prestado por los dispositivos **IoT**, y el alcance también se ve limitado por los mismos factores. Como la mayoría de estos dispositivos remotos son pequeños, y suelen funcionar con baterías, las opciones de red que requieren mucha energía, como el uso de redes celulares o satelitales, tampoco son ideales para la mayoría de los escenarios.

Beneficios de las redes mesh para redes IoT

Las **redes mesh** pueden aportar muchas ventajas cuando son usadas en redes de dispositivos **IoT**, y la más destacada es la versatilidad de la misma. Cuando se utiliza una topología de malla, añadir un nuevo nodo sólo requiere poner el nuevo dispositivo directamente en el área dentro del alcance de la red, tan simple como eso. La capacidad y el alcance de la red se amplia sin introducir nuevas conexiones. Esto es debido a la característica de auto-configuración de la red que logrará expandirse automáticamente. Por lo tanto, la estructura de la red sería más sencilla, y el precio de cubrir una zona más amplia, especialmente en zonas rurales sin una red fiable en todas partes podría ser mucho mejor. El consumo de energía también puede reducirse significativamente ya que los dispositivos solo puede conectarse al dispositivo de la **red mesh** más cercano en lugar de hacerlo a un centro de red central distante. Esta conexión puede formar una cadena de nodos, reduciendo el costo de energía para cubrir un área mas distante.

Además de las ventajas en términos de escalabilidad y reducción de costos al establecer una nueva red, las **redes mesh** también proporcionan una topología más robusta para todo tipo de aplicaciones en el mundo del **IoT** cuando se trata de eventos poco probables como los desastres naturales por ejemplo. Incluso cuando alguna parte del enlace se destruye y varios dispositivos quedan fuera de la conexión. El resto de la red tiene la

habilidad de seguir auto-conectada y auto-configurada en una nueva red y en perfecto funcionamiento. En otras topologías de redes más convencionales este problema conllevaría un corte total del servicio.

Desventajas de las redes mesh para redes IoT

Ciertamente y como era de esperarse, las **redes mesh** tienen algunas desventajas. La estructura no convencional de la misma requiere un nuevo soporte del protocolo de red y el protocolo debe ser compatible con la red existente, ya que los dispositivos **IoT** se conectaran eventualmente a la internet. Además, la reparación de la red cuando se produce un apagón de mayor escala podría ser mas difícil, ya que el corte de la conexión difícilmente puede ser detectado cuando se desconecta de una red más grande. Sin embargo, esto se puede resolver introduciendo un informe de verificaciones de errores en los paquetes, lo que se expresa en tiempos mas lentos. Por último, el retraso y la escalabilidad de la red también están limitados por su naturaleza. Esto puede no ser un problema en los dispositivos **IoT**, pero debe tenerse en consideración cuando se trata de seleccionar herramientas apropiadas para el trabajo requerido.

ANÁLISIS DE MERCADO

El concepto de **internet de las cosas** implica extender la conectividad a internet más allá de los dispositivos estándares, como computadoras de escritorio, portátiles, teléfonos inteligentes y tabletas, a cualquier otra gama de dispositivos físicos tradicionalmente llamados “tontos”. Esto se debe a que no están habilitados a internet como lo son los objetos cotidianos. Incorporando esta tecnología, esta clase de dispositivos pueden conectarse e interactuar a través de internet, y pueden ser monitoreados y controlados remotamente lo que nos abre un abanico enorme de posibilidades y, por ende, productos que pueden ser comercialmente rentables.

Los dispositivos conectados forman parte de un escenario en el que cada dispositivo habla con otros en un entono para automatizar las tareas domesticas e industriales, y para comunicar los datos de los sensores a los usuarios, empresas y otras partes interesadas. Los dispositivos **IoT** están pensados para funcionar en conjunto ya sea para la persona en su día a día, el hogar, la industria o en las empresas. Como tales, los mismos pueden clasificarse en tres grupos muy notorios: consumidor, empresa e industria.

Los dispositivos de consumo incluyen televisores inteligentes, altavoces inteligentes, juguetes, artículos de vestir y electrodomésticos inteligentes. Los medidores inteligentes, sistemas de seguridad comercial y las tecnologías de ciudades inteligentes, como las que se usan para monitorear tráfico y condiciones climáticas son ejemplos de dispositivos **IoT** industriales y empresariales. Otras tecnologías, como el aire acondicionado inteligente, los termostatos inteligentes, iluminación inteligente y seguridad inteligente abarcan el uso domestico, empresarial e industrial debido a que son vistos y usados en todos los sectores.

En un hogar inteligente, por ejemplo, el usuario llega a su casa y su coche se comunica con el garaje para abrir la puerta automáticamente. Una vez adentro, el termostato ya esta ajustado a su temperatura preferida, y la iluminación ajustada a una intensidad menor y un color elegido que reflejen relajación, ya que los datos de su marcapaso indican que ah sido un día estresante.

En las empresas, los sensores inteligentes ubicados en una sala de conferencias pueden ayudar a un empleado a localizar y programar una habitación disponible para una reunion, asegurando que el tipo de sala, tamaño y características sean adecuadas y estén disponibles para tal fin. Cuando los integrantes de la reunion entran a la sala, la temperatura se ajusta de acuerdo con la cantidad de personas que haya, y las luces se atenúan a medida que las presentaciones en la pantalla van pasando y el orador comienza su presentación.

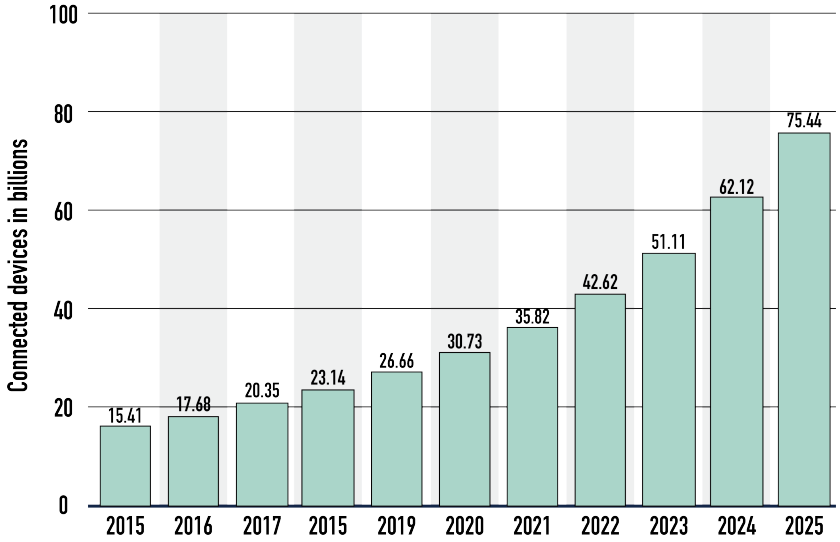


Imagen Nº8: Cantidad de productos IoT

En una planta, una maquina de la linea de montaje equipada con sensores proporciona datos de los sensores al operador de la plata, informándole de cualquier anomalía y prediciendo cuando será necesario sustituir las piezas o hacer algún tipo de mantenimiento. Esta información puede prevenir paradas inesperadas, junto con la perdida de productividad y generación de beneficios.

En la practica, todo este tipo de notificaciones pueden alertar a los usuarios de lo que esta mal, así como de las piezas necesarias para solucionar un problema, evitando la necesidad de enviar a un trabajador de mantenimiento para diagnosticar un problema.

La gestión de dispositivos **IoT** ayuda a las empresas a integrar, organizar, supervisar y administrar a distancia los dispositivos habilitados a una conexión a internet a gran escala, ofreciendo características fundamentales para mantener la salud, la conectividad y la seguridad de los mismos a lo largo de todo su ciclo de vida util. Estas características incluyen:

- Registro de dispositivos
- Autenticación/autorización del dispositivo
- Configuración del dispositivo
- Aprovisionamiento de dispositivos
- Monitoreo y diagnostico

- Solución de problemas

Hoy en día es un hecho que el numero de productos **IoT** haya superado nada más ni nada menos que a la cantidad de humanos en el planeta. Aproximadamente hay alrededor de 7.62 mil millones de seres humanos en el planeta tierra, pero para su sorpresa, para el año 2021 el gráfico creciente de dispositivos **IoT** nos muestra que puede haber un estimado de 20 mil millones de dispositivos en funcionamiento con un fuerte aumento debido a la llegada de las redes 5G.

Si se hace un promedio, en algunos años cada individuo de America del Norte tendría más de 10 productos **IoT**. El número de dispositivos conectados a Internet que tiene la gente está aumentando, especialmente en Estados Unidos.

Según el ultimo indice anual de redes visuales previsto por **Cisco**, habra cuatro dispositivos conectados a la red por persona en todo el mundo para el año 2021. Sin embargo, en América del Norte habrá 13 dispositivos y conexiones en red por persona. Esto significa que, además de los teléfonos inteligentes y los televisores conectados, los consumidores adoptarán muchos más aparatos.

Estados Unidos está muy por encima de la media por región cuando se trata de dispositivos **IoT**. Por ejemplo, a continuación se muestran los números proyectados de dispositivos conectados por persona para el 2021:

- América del Norte →13

- Europa Occidental → 9

- Europa Central y Oriental → 4

- América Latina → 3

- Asia y el Pacífico → 3

- Oriente Medio y África → 1

Durante el mismo período de tiempo que la adopción de productos **IoT** aumenta masivamente, las velocidades de banda ancha casi se duplicarán. El aumento en velocidad no significa necesariamente que los consumidores harán las cosas más rápido. Sin embargo, sí significa que el acceso de los consumidores a la información y al contenido, especialmente a la transmisión de video, será accesible más rápidamente a través de más dispositivos.

Con el tiempo, es probable que los consumidores se apoyen más en sus dispositivos inteligentes para automatizar las tareas para ellos.

¿Cómo funciona el internet de las cosas y qué convierte a un dispositivo en inteligente?

Depende básicamente de dos cosas para poder transformar un dispositivo normal en uno **IoT**:

1- El dispositivo tiene que tener la capacidad de conectarse a internet de cualquier manera, ya se directa o indirectamente.

2- El dispositivo debe integrar sensores, *software* funcional, actuadores o alguna tecnología que soporte conexión a la red. No tiene que darse todos juntos necesariamente.

Cuando ambas funcionalidades se combinan, se forma un dispositivo **IoT**. Hace algunos años se usaban simples relojes para ver la fecha y la hora, ahora los relojes inteligentes permiten al usuario ver el ritmo cardíaco, las calorías quemadas, pasos que se han dado, leer los mensajes que recibe tu teléfono inteligente, entre muchas otras.

El mercado de los productos **IoT** se está expandiendo rápidamente día a día y también se está haciendo más popular con el drástico aumento del número de usuarios que los usan diariamente.

Ventajas de los dispositivos de IoT

Existen varias ventajas que vuelven al internet de las cosas como una tecnología tan interesante y de continuo crecimiento en estos tiempos.

- Proporciona automatización y control tanto en hogares como en industrias y empresas.

- Integrado con información técnica, por lo que es mejor para operar.

- Posee una fuerte característica de monitoreo.

- Ahorra mucho tiempo.

- Ayuda a ahorrar más dinero reduciendo la tarea manual y el tiempo.

- Permite automatizar tareas del día a día.

- Incrementa la eficiencia.

Desventajas

Aunque existen varias ventajas, también hay ciertas desventajas.

- Los dispositivos de Internet de las cosas no tienen ningún estándar de compatibilidad internacional aunque esto esta cambiando poco a poco.

- Pueden llegar a ser muy complejos y resultar en una falla.

- Pueden verse afectados por la violación de la privacidad y la seguridad.

- La seguridad de los usuarios se ve reducida.

- Reducción en el empleo de tareas manuales, lo que resulta en una reducción de puestos de trabajo.

- Los dispositivos **IoT** pueden tomar el control de la vida a su debido tiempo con el aumento de la tecnología de la IA.

Productos IoT existentes

Como se dijo anteriormente existe una enorme cantidad de productos de internet de las cosas los cuales se pueden comprar ahora mismo. Estos abarcan un gran espectro que incluye desde asistentes personales hasta robots de entretenimiento inteligente pasando por detectores de humo, sensores de pureza del aire, cerraduras inteligentes y muchos más.

A continuación se muestran los 10 productos **IoT** mas populares en 2020 según "**ROBOTS.NET**":

1. Amazon Echo Plus

Imagen Nº9: Amazon Echo Plus

Código Nº1: Amazon Echo Plus

Amazon Alexa Echo Plus resulta ser uno de los dispositivos **IoT** más celebres que ha lanzado el gigante de la tecnología **Amazon**. Como su nombre lo sugiere, es conocido por ser un altavoz inteligente capaz de reproducir audio 360°. Además, viene dotado de una interfaz de control de temperatura y un *hub* para teléfonos inteligentes.

2. Nest Learning Thermostat

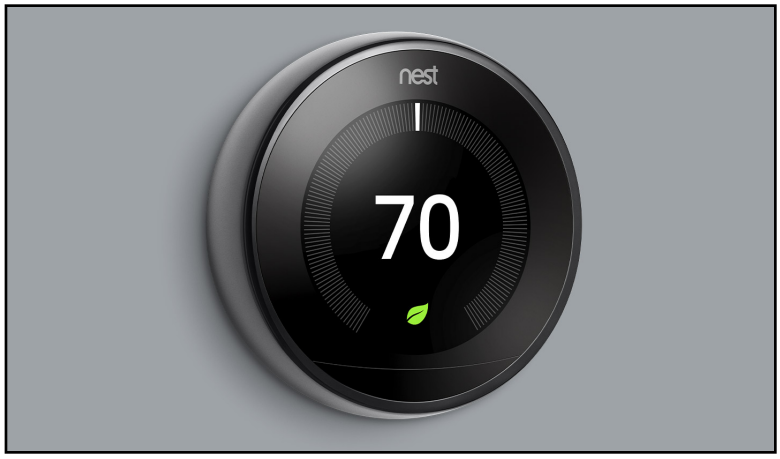


Imagen Nº10: Nest Learning Thermostat

Código Nº2: Nest Learning Thermostat

El **Nest Learning Thermostat** (Nest termostato de aprendizaje) es un termostato inteligente electrónico, programable y de auto-aprendizaje desarrollado por la empresa **Net Labs** que optimiza la calefacción y refrigeración de hogares y negocios para poder así reducir el consumo energético.

3. Kuri Mobile Robot

Imagen Nº11: Kuri Mobile Robot

Código Nº3: Kuri Mobile Robot

Kuri Mobile Robot es un robot doméstico diseñado para interactuar tanto con el usuario como con su familia y captar fotos y videos del día a día. Tiene una personalidad expresiva y su propio y único lenguaje robótico. El amigable robot cuenta con WiFi,

Bluetooth, ruedas, cámara full HD, sensores, micrófonos y parlantes, cada centímetro de **Kuri** está diseñado con el único objetivo de hacer a este pequeño robot un ayudante adorable.

4. Netgear Orbi WiFi System

Código Nº4: Netgear Orbi WiFi System

Imagen Nº12: Netgear Orbi WiFi System

Netgear Orbi WiFi System es un dispositivo bastante poderoso que cubre un enorme rango de más de 450 metros cuadrados. Además, incluye un enrutador y un sistema de satélites que puede ser conectado con su actual ISP. Por lo tanto, es bastante evidente que este producto tiene el potencial de eliminar todos los problemas relacionados con el alcance de la red WiFi hogareña y puede conectarse en cuestión de minutos. Es el producto idea para los casos en los cuales la red actual no logra cubrir la totalidad del hogar, oficina o cualquiera sea el lugar.

5. Samsung SmartThings Hub

Código Nº5: Samsung SmartThing Hub

Imagen Nº13: Samsung SmartThing Hub

La mayoría de los dispositivos **IoT** sólo sirven para un problema específico. Sin embargo, el fabricante coreano **Samsung** ha ido más allá de estos enfoques tradicionales y toma el control de todo el hogar en una sola unidad. Ya sea el termostato, las luces, los enchufes, altavoces o cualquier otra cosa con las especificaciones técnicas requeridas, todo puede ser conectado a un centro que también puede ser controlado desde el teléfono móvil del usuario.

6. Petnet Smart Pet Feeder

Imagen Nº14: Petnet Smart Pet Feeder

Código Nº6: Petnet Smart Pet Feeder

Petnet Smart Pet Feeder es un dispensador de comida para mascotas inteligente, con su ayuda, el usuario ya no tiene que quedarse en su casa con la mascota todo el tiempo. Esta tecnología puede determinar el mejor alimento para la mascota, el momento del día y la cantidad de comida. La mejor parte de este producto es que el propietario puede cuidar su mascota incluso sin estar en el hogar y permite dispensar comida como uno mismo lo haría.

7. Philips Hue Bulb System

Imagen Nº15: Philips Hue Bulb System

Código Nº7: Philips Hue Bulb System

De entre todas las plataformas de iluminación inteligente que existen al día de hoy, Hue es la más establecida, la más desarrollada y la que cuenta con mejor conexión y fue furor en ventas en el año 2019. No importa el tipo de usuario, este sistema tiene perfecta asociación con asistentes personales como **Alexa**, **Google Assistant**, o para los apasionados de **IFTTT** ("*If This Then That),* un conocedor de **Logitech Harmony**, por solo nombrar alguna plataformas.

8. Nest Smoke Alarm

Imagen Nº16: Nest Smoke Alarm

Código Nº8: Nest Smoke Alarm

Nest Smoke Alarm es una solución bastante robusta diseñada para detectar humo y monóxido de carbono en ambientes residenciales como hogares u oficinas. Consta de multiples sensores que le ayudan a comprender lo que esta sucediendo en la sala donde se encuentra. Por otro lado esta alarma de humo esta dotada de un sensor fotoeléctrico para detectar incendios lentos. Toda esta información es recibida como notificación en el teléfono móvil del usuario que lo posea.

9. Flow – Air Pollution Monitor

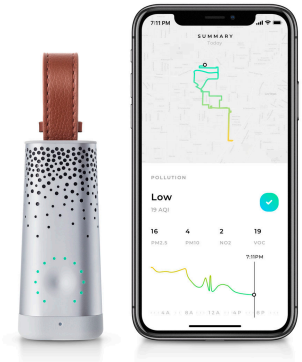


Imagen Nº17: Flow – Air Pollution Monitor

Código Nº9: Flow – Air Pollution Monitor

Air Pollution Monitor es uno de los inventos más innovadores de la industria, es el primer sensor ambientar de su clase en un tamaño tan reducido, tan portátil y tan personal. Este dispositivo monitorea la calidad del aire, predice la contaminación y planifica los desplazamientos del usuario con solamente el sensor personal y su aplicación complementaria.

10. August Smart Lock

Imagen Nº18: August Smart Lock

Código Nº10: August Smart Lock

August Smart Lock es un dispositivo que nos permite cerrar y abrir la puerta desde cualquier lugar como así también dar llaves digiérales a invitados, todo esto controlado desde un teléfono inteligente. Todas las cerraduras inteligentes August se conectan al cualquier cerrojo existente en el hogar del lado interior de la puerta para que el usuario pueda seguir usando las llaves normalmente.

Productos Mesh existentes

Como se pudo ver anteriormente existe una extensa y casi infinita lista de productos **IoT** en el mercado actual y con tendencia creciente hacia muchos más pero, contrario a esto, los productos que usan la topología mesh son pocos o casi nulos, ni hablar de la combinación de ambos.

A fecha de hoy, año 2020 las redes mesh son usadas exclusivamente en productos WiFi para poder extender el alcance de los routers existentes en los hogares los cuales se puede decir que no tienen una distancia de señal elevada. Es muy importante mantener un buen rendimiento de la red de internet para, por ejemplo, juegos en línea, transmisión de video, dispositivos domésticos inteligentes, entre otros. También hay que tener en cuenta el elevado aumento en el trabajo de forma remota lo que vuelve muy importante a las aplicaciones de trabajo y los diferentes modos de comunicación, especialmente las

videoconferencias. Es aquí donde una cobertura completa de la casa se convierte en algo más que un bien de lujo.

Muchos de los routers inalámbricos existentes pueden proporcionar una fuerte cobertura a la mayoría de las habitaciones de una casa típica de tamaño medio, pero las casas más grandes y viviendas con paredes densas, varios pisos, subestructuras de metal y hormigón y otros impedimentos estructurales pueden requerir componentes adicionales para llevar la red WiFi a zonas que el router no puede alcanzar por si solo. Los extensores de alcance hace un muy buen trabajo en rellenar las zonas muertas, pero normalmente solo proporcionan la mitad del ancho de banda que se obtiene del router principal. Por otro lado, los puntos de acceso ofrecen más ancho de banda que los extensores de alcance, pero requieren una conexión por cable con el router. Cabe aclarar que ambas soluciones típicamente crean un nuevo SSID de red al que el usuario tiene que acceder cuando se mueve durante las diferentes zonas.

¿Qué es un sistema WiFi Mesh?

Estos sistemas están diseñados para cubrir todo el área del hogar con acceso inalámbrico a la red, los **sistemas WiFi mesh** son una especie de híbrido, compuesto por varios componentes de red. Por un lado cuentan con un enrutador principal que se conecta directamente al modem, y una serie de módulos satélites, o nodos, que se colocarán por todo el complejo. Todos estos componentes forman parte de una única red inalámbrica y comparten el mismo SSID y contraseña. A diferencia de los extensores de alcance, que se comunican con el router a través de las bandas de radio de 2,4GHz o 5GHz, la mayoría de los nodos de estos sistemas WiFi utilizan la **tecnología mesh** para “hablar” con el router y entre si.

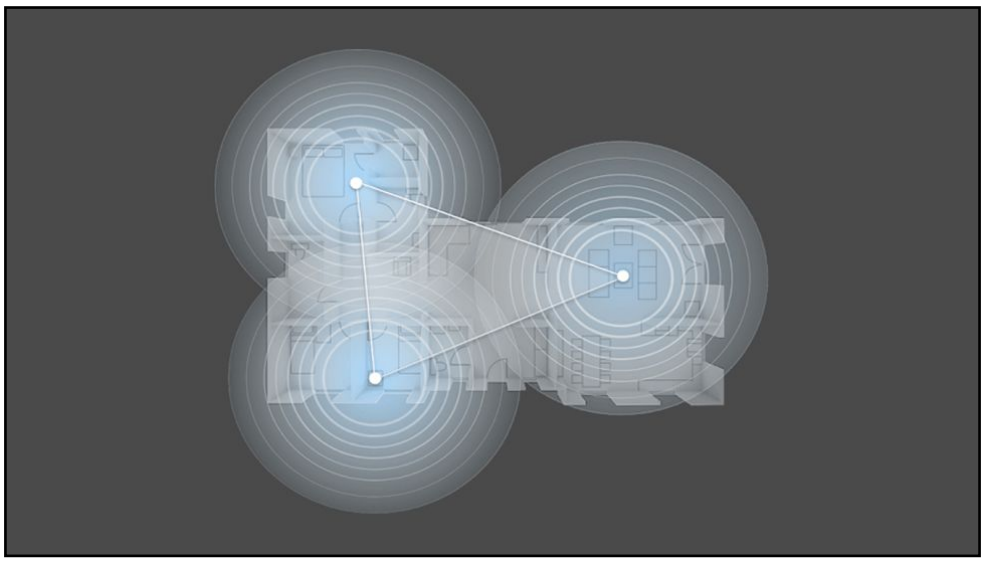


Imagen Nº19: Sistema WiFi Mesh

Cada nodo sirve como punto de salto para otros nodos del sistema. Esto ayuda a los nodos más alejados del enrutador a entregar una fuerte señal WiFi ya que están hablando con otros nodos y no dependen de las comunicaciones uno a uno con el enrutador.

Los **sistemas WiFi mesh** son muy fáciles de ampliar y actualmente no cuentan con un numero límite de nodos que se puedan añadir. También son fáciles de gestionar por medio de un teléfono inteligente, lo que permite desactivar el acceso a la red para dispositivos específicos y dar prioridad de conexión a determinados dispositivos sin tener que acceder a una complicada consola de red.

Productos WiFi mesh existentes

A continuación se muestran los 10 **sistemas de WiFi mesh** mas populares en 2020 según "**T3.COM**":

1. EERO WiFi Mesh x

Imagen Nº20: EERO WiFi Mesh

Código Nº11: EERO WiFi Mesh

2. Netgear Orbi WiFi 6

Imagen Nº21: Netgear Orbi WiFi 6

Código Nº12: Netgear Orbi WiFi 6

3. TP-Link Deco P9

Imagen Nº22: TP-Link Deco P9

Código Nº13: NTP-Link Deco P9

4. Google Nest WiFi

Imagen Nº23: Google Nest WiFi

Código Nº14: Google Nest WiFi

5. Tenda MW6 Nova

Imagen Nº24: Tenda MW6 Nova

Código Nº15: Tenda MW6 Nova

6. BT Whole Home

Imagen Nº25: BT Whole Home

Código Nº16: BT Whole Home

7. TP-Link Deco X60

Imagen Nº26: TP-Link Deco X60

Código Nº17: TP-Link Deco X60

8. Zyxel Multy X Whole Home WiFi Mesh System

Imagen Nº27: Zyxel Multy X

Código Nº18: Zyxel Multy X

9. Linksys Velop

Imagen Nº28: Linksys Velop

Código Nº19: Linksys Velop

10. Sky Q

Imagen Nº29: Sky Q

Código Nº20: Sky Q

Productos IoT Mesh existentes

Anteriormente se vieron las ventajas y desventajas de las **redes mesh** como así también de los productos **IoT** y es cuando surge la pregunta, ¿Porque no unir estos dos términos en un único producto? La respuesta puede sorprender y es que en la actualidad no existen muchos productos **IoT** que utilicen **redes mesh**. Entre los más comunes se encuentran Zigbee y Zware pero estos no son los unicos ya que existe un nuevo producto con un enforque claro hacia este tipo de redes.

El nombre de este producto podríamos decir que no es muy innovador pero si muy representativo y es llamado simplemente "***MESH***". Este sistema se basa en bloques, cada uno puede ser un sensor o un periférico con funciones incorporadas para facilitar la creación de prototipos y la construcción de proyectos para internet de las cosas.

Como se puede apreciar en **Imagen Nº30** este producto cuenta con 7 tipos de bloques distintos que van desde un led hasta sensores de movimiento pasando por accionadores y

entradas y salidas digitales. Todos estos nodos generan una **red mesh** por la cual se comunican los unos con los otros como así también lo hacen con la aplicación nativa.

Imagen Nº30: Bloques MESH

Mesh cuenta con una aplicación muy pulida y de muy fácil manejo la cual se basa en codificación visual tal que simplemente hay que conectar los iconos con cables y eso es todo. No se necesita entrenamiento técnico o avanzado para usar este sistema debido a la simpleza del mismo. **MESH** App no solamente cuenta con bloques de *hardware* que representan los módulos físico sino que también puede hacer uso de bloques de *software* los cuales pueden ser lógicos, dispositivos y extensiones. Los bloques de hardware aparecerán en la aplicación al momento en el que el dispositivo físico logre vincularse a la red mesh sobre la que se trabaja y es ahí cuando esta listo para ser usado.

Tipos de **bloques Mesh** de hardware dentro de la aplicación:


Bloque de botón


Bloque de LED


Bloque de movimiento


Bloque de presencia


Bloque de brillo


Bloque de temperatura y humedad


Bloque de GPIO

Tipos de bloques de software dentro de la aplicación:

Dispositivos	Logicos	Expansiones
Bloque de cámara	Bloque AND	Bloque Philips Hue
Bloque de micrófono	Bloque Timer	Bloque Gmail
Bloque de altavoz	Bloque Switch	Bloque IFTTT
Bloque de notificación	Bloque Contador	Bloque OLYMPUS AIR
Bloque de música		

Tabla Nº1: Bloques de software

Cada dispositivo inteligente comienza con un prototipo, y **MESH** GPIO es una simple interfaz para placas de desarrollo como **Arduino** y **Raspberry Pi** o actuadores como puede ser un motor de corriente continua. Este sistema permite la integración con cualquier dispositivo inteligente o servicios web como **IFTTT**, **Google Assistant**, **Amazon Alexa**, **Twitter**, **Google Sheets**, **Philips Hue**, **Nest**, **LIFX**, **WeMo** y otras 350 plataformas más.

Como se mostró anteriormente, **MESH** cuenta con 7 bloques físico que pueden adquirirse en este momento en su página web y se los detallará a continuación:

MESH Botón:

MESH botón es un pulsador diseñado para la creación rápida de prototipos y el armado instantáneo de proyectos. El botón **MESH** puede ser programado para activar acciones como encender una bombilla inteligente, enviar un mensaje de texto, tomar una foto con un teléfono, encender un motor o registrar un evento en una hoja de cálculo.

MESH LED:

MESH LED es un indicador LED multicolor, diseñado para la creación rápida de prototipos y el armado instantáneo de proyecto. El bloque puede controlar el color y patrón del indicador luminoso como así también cambiar su brillo, duración, ciclo e intervalo.

MESH Movimiento:

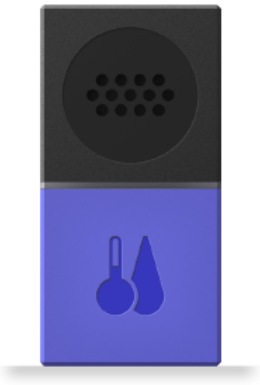
MESH movimiento es un acelerómetro diseñado para creación rápida de prototipos y el armado instantáneo de proyectos. El bloque puede detectar sacudidas, volteos, vibraciones y orientación.

MESH Presencia:

MESH presencia es un sensor de movimiento infrarrojo diseñado para creación rápida de prototipos y el armado instantáneo de proyectos. Cuanta con un sensor de movimiento infrarrojo y puede usarse para construir proyectos de movimiento controlado cuando se detecta movimiento.

MESH Brillo:

MESH brillo es un sensor de luz diseñado para creación de aplicaciones que detectan luz ambiental. Es un sensor de luz ambiental que puede ser usado cuando se detecta la luz ambiental en un cierto rango o incluso cambios en el brillo de la luz.



MESH Temperatura y Humedad:

MESH temperatura y humedad es un sensor diseñado para aplicaciones que detectan cambios en la temperatura y humedad. Puede ser usado para crear proyectos de temperatura y humedad controlados como un ventilados o aire acondicionado. Este bloque incluye ajustes personalizadles para detectar, comprobar y percibir los cambios de temperatura y humedad.

MESH GPIO:

MESH GPIO es un bloque de 10 pines de entradas/salidas de propósito general para la creación rápida de prototipos y el armado de proyectos.

Como toda cosa buena tiene que tener algo de malo y en este caso una muy grande, el precio. El sistema **MESH** tiene un muy elevado valor para lo que da a cambio. Cada uno de los bloques cuesta entre 40U$D y 60U$D lo cual, si se hace una suma es una muy alta cantidad de dinero para tener un único set completo de bloques, sin repetir ninguno.

Esto hace ver que un muy buen producto se puede volver inalcanzable para la mayoría de los compradores, en un mundo en donde un parlante inteligente cuesta el mismo valor y da a cambio mucho más, o una lampara inteligente se puede conseguir por un cuarto del precio, solo por dar algunos ejemplos.

TECNOLOGÍAS UTILIZADAS

Los tecnologías y herramientas de diseño de sistemas embebidos se ven en el día a día, existe una multitud de entornos de trabajo, *frameworks*, librerías, arquitecturas y sistemas tanto de pago como *software* libre. A esto hay que sumarte que este trabajo final no solo está compuesto por cuestiones de código sino también de diseño e implementación de *hardware* por lo que las herramientas y tecnologías usadas son aún más.

El diseño y desarrollo de proyectos que integran el *hardware* y el *software* se basa en una constante decisión de que usar y que no dependiendo de el tiempo en el que se quiera llevar a cabo el trabajo, el alcance del producto, la *performance*, habilidades, costos, entre otras.

A continuación se detallarán las herramientas y tecnologías más importantes usadas en el desarrollo de este proyecto final. Si bien no son las únicas existentes si son las que mejor se amoldan a las características de este trabajo por las cuestiones antes mencionadas.

C++

C++ es un lenguaje de programación de uso general creado por *Bjaner Stroustrup* como una extension del lenguaje de programación **C**, también es llamado "**C** con clases". El lenguaje se ah expandido significativamente con el tiempo, y el **C++** actual tiene características orientadas a objetos, genéricas y funcionales, ademas de facilidades para la manipulación de memoria de bajo nivel. Casi siempre se implementa como un lenguaje compilado, y muchos proveedores proporcionan compiladores de **C++**, incluyendo la **Fundación de Software Libre**, **LLVM**, **Microsoft**, **Intel**, **Oracle**, e **IBM**, por lo que está disponible en muchas plataformas.

C++ fue diseñado hacia la programación de sistemas, con rendimiento, eficiencia y flexibilidad de uso o también para sistemas embebidos donde los recursos son limitados. "**C** con clases" también se ha encontrado útil en muchos otros contextos, siendo sus principales puntos fuertes la infraestructura de *software* y las aplicaciones con recursos

limitados, incluyendo aplicaciones de escritorio, videojuegos, servidores y aplicaciones de rendimiento crítico.

C++ está normalizado por la Organización Internación de Normalización (ISO), y la última versión de la norma fue ratificada y publicada por la ISO en diciembre de 2017 como ISO/CEI 14882:2017 (también conocida como **C++**17).

Como vimos, **C++** puede ser usado en sistemas embebidos para la programación de microprocesadores como es el caso de uso en este trabajo, a este lenguaje se lo llama "***Embedded* C++**" o "**C++ embebido**". El mismo fue definido por un grupo industrial encabezado por los principales fabricantes japoneses de unidades centrales de procesamiento (CPU), entre ellos **NEC**, **Hitachi**, **Toshiba**, etc, para abordar las deficiencias de **C++** en aplicaciones embebidas. El objetivo es preservar las características orientadas a objetos más útiles del lenguaje y, al mismo tiempo, minimizar el tamaño del código, maximizar la eficiencia de la ejecución y simplificar la construcción del compilador.

ESP8266

ESP8266 es un microchip WiFi de bajo costo, con un *stack* completo de TCP/IP y capacidad de microcontrolador, producido por la empresa china **Espressif Systems**.

El chip llego por primera vez a manos de los fabricantes occidentales en agosto del año 2014 con el módulo **ESP-01**, fabricado por **Ai.Thinker**. Este pequeño módulo permite a los microcontroladores conectarse a una red WiFi y hacer conexiones TCP/IP simples usando comandos del estilo de *Hayes*. Sin embargo, al principio casi no había documentación en inglés a cerca del chip y ni de los comandos que aceptaba.

El muy bajo precio que tenía este módulo como así también el hecho de que había muy pocos componentes externos en el, sugería que, eventualmente, podría ser muy barato de fabricar en altos volúmenes. Esto es lo que atrajo a muchos "*hackers*" a explorar por dentro el módulo, el chip y el software que tenía adentro, como así también a la traducción del chino de la documentación.

El **ESP-12F** es un módulo WiFi basado en el **ESP8266**, con una memoria *Flash* de 32Mbit incorporada, todo esto se encuentra embebida en un encapsulado SMD22. También cuenta con una antena PCB y una carcasa metálica. Este trabajo final hace uso del **ESP-12F** como microcontrolador principal de los módulos.

Principales *características*:

- Micro MCU de 32 bit de ultra bajo consumo.
- Corre frecuencias de 80MHz y 160MHz como así también soporte para RTOS.
- Conversor ADC de alta precisión de 10 bits integrado.
- Soporte para UART/GPIO/ADC/PWM/I2C.
- Encapsulado SMD22.
- WiFi MAC/ BB/RF/PA/LNA incorporado.
- Soporte de múltiples modos *sleep*.
- Soporte de actualización de *firmware* local y remoto (UART y FOTA).

Arduino

Arduino es el ecosistema de *hardware* y *software* de código abierto líder en el mundo. La empresa ofrece una gama de herramientas de *software*, plataformas de *hardware* y documentación que permiten a casi todo el mundo ser creativo con la tecnología.

A su vez, **Arduino** es una herramienta muy popular para el desarrollo de productos **IoT**, así como una de las herramientas más exitosas para la educación. Cientos de miles de diseñadores, ingenieros, estudiantes, desarrolladores y constructores de todo el mundo la utilizan para innovar en áreas tales como los hogares inteligentes, la agricultura, los vehículos autónomos, los juegos, etc. **Arduino** es el primer proyecto de *hardware* de código abierto de gran alcance y se creo para construir una comunidad que pueda ayudar a difundir el uso de la herramienta y beneficiarse de las contribuciones de cientos de

persona que ayudan a depurar el código, escribir ejemplos, crear tutoriales, apoyar a otros usuarios foros y construir miles de grupos en todo el mundo.

Desde la fundación de este proyecto, se han introducido muchos nuevos tableros de desarrollo y bibliotecas de *software*, ampliando la gama de posibilidades disponibles para la comunidad. Al día de hoy, más de una década después, **Arduino** sigue proporcionando *hardware* y *software* de código abierto para dar vida a nuevas ideas. La apertura y facilidad de uso es lo que ha llevado a la adopción pasiva de proyectos electrónicos basados en microcontroladores y fue un catalizador en la creación del *Maker Movement*. **Arduino** se ha convertido en la opción número uno para los fabricantes de electrónica, especialmente para el desarrollo de soluciones para el mercado de **IoT**.

KiCad

KiCad es un paquete de software libre para la automatización del diseño electrónico (EDA por sus siglas en ingles “Electronic Design Automation”). Facilita el diseño de esquemas para circuitos electrónicos y su conversión al diseño de PCB. Dentro de este paquete existen herramientas para crear una lista de materiales, material gráfico, archivos Gerber y vistas en 3 dimensiones del PCB y sus componentes.

KiCad fue creado en 1992 por Jean-Pierre Charras mientras trabajaba en el **IUT** de Grenoble. Desde entonces, **KiCad** ha ganado un numero de contribuyentes tanto voluntarios como pagados. En particular, en 2013 la selección BE-CO-HT del **CERN** comenzó a aportar recursos para ayudar a fomentar el desarrollo de hardware abierto ayudando a mejorar **KiCad** para que esté a la par con las herramientas comerciales de EDA.

KiCad hace uso de un entorno integrado para todas las etapas del proceso de diseño: Captura de esquemas, diseño de PCB, generación/visualización de archivos Gerber y edición de bibliotecas.

JLC PCB

JLCPBC es una empresa líder mundial en la fabricación de prototipos de PCB y un fabricante de tecnología especializado en la producción rápida de prototipos de PCB y pequeños lotes de los mismos.

Con mas de 14 años de experiencia en la fabricación de placas electrónicas, el fabricante chino **JLCPCB** tiene más de 800.000 clientes, con más de 18.000 pedidos en línea de prototipos y producción de pequeñas cantidades por día. La capacidad de producción mensual es de 600.000 metros cuadrados para varios tipos de PCB de una, dos y varias capas. **JLCPBC** es un fabricante profesional de placas que se caracteriza por su gran escala, su equipamiento sobresaliente, su gestión estricta y su calidad superior como así también pequeños precios.

El tiempo de construcción significa el tiempo de producción de la misma. Es decir, el tiempo desde el inicio de la producción física hasta la producción termina y lista para entregar. Estos tiempos varían dependiendo de la cantidad, la complejidad de sus diseños y los procesos de ensamblaje si son necesarios.

JLCPBC no solo produce PCB sino que también se puede pedir el stencil en el caso que se quiera hacer la producción de las placas en, por ejemplo, una máquina pick and place. El mismo consiste de una plancha de aluminio con la forma de los pads correspondiente a cada uno de los componentes. Por otro lados cabe aclarar que al momento de generar el pedido, la página da la opción de elegir uno de los colores de PCB, esto da una flexibilidad y un extra a la hora de elegir un fabricante de placas electrónicas.

Atom

Atom es un editor de texto y código fuente gratuito y de código abierto para macOS, Lunuz y Microsoft Windows con soporte para plug-ins escritos en **Node.js**, y control **Git** embebido, desarrollado por **GitHub**. Atom es una aplicación de escritorio construida usando tecnologías web, la mayoría de los paquetes de extensiones tienen licencia de *software* libre y están construidos y mantenidos por la comunidad. Este editor de código esta basado en **Electron** (antes conocido como **Atom Shell**), un marco de trabajo que permite aplicaciones de escritorio multi-plataforma usando **Chromium** y **Node.js**, está escrito en CoffeeScript y Less.

Características:

- Edición multi-plataforma, Atom funciona en diferentes sistemas operativos. Ya sea macOS, Windows o Linux.
- Gestor de paquetes incorporado, busca e instala nuevos paquetes o crea paquetes propios, todo desde el mismo editor.
- Autocompletado inteligente, Atom ayuda a escribir código más rápido con un autocompletado inteligente y flexible.
- Múltiples panales, divide la interfaz en múltiples paneles para comparar y editar el código en ambos archivos al mismo tiempo (multi tarea).
- Busca y reemplaza, Atom permite encontrar, personalizar y reemplaza texto mientras se escribe en un archivo o en todos los proyectos.

Visual Studio Code

Visual Studio Code

Visual Studio Code es un editor de código fuente gratuito hecho por **Microsoft** para Windows, Linux y macOS. Sus características incluyen soporte para depuración, resaltado de sintaxis, finalización inteligente de código, fragmentos y **Git** incorporado. Los usuarios pueden cambiar el tema, los ajados de teclado, las preferencias, e instalar extensiones que añaden funcionalidades adicionales.

El código fuente de **Visual Studio Code** proviene del proyecto **VSCode** de software libre y de código abierto de **Microsoft**, publicado bajo la permisiva Licencia Expat, pero el código binario compilado es gratuito para cualquier uso. En la cuesta de desarrolladores Stack Overflow 2019, **VSC** fue clasificado como la herramienta de entorno de desarrollo más popular, con el 50,7% de los encuestados, al día de hoy es el editor de texto por referencia en el desarrollo de software.

Características:

- IntelliSense, es un término generalmente usado para una variedad de características de edición de código tales como finalización de código, información de parámetros, información rápida, finalización de código, asistencia de contenido y sugerencia de código.

- Paleta de comandos, **Visual Studio Code** consta de una terminal integrada, empezando por la raíz del proyecto. Esta característica es muy conveniente ya que no hay que cambiar de ventana o alterar el estado de una terminal existente para realizar una tarea rápida de línea de comando.

- Control de versiones integrado, **VSCode** tiene la integración con **Git** incorporada, lo que hace muy fácil ver instantáneamente los cambios que está habiendo en el proyecto.

- Depuración, una de las características claves de este editor de texto es su gran soporte de depuración. El depurador incorporado ayuda a acelerar la edición, compilación y depuración de bucles.

- Edición lado a lado en diferentes archivos.

PlatformIO

PlatformIO es una herramienta profesional multiplataforma, multiarquitectura y multimarco diseñada para ingenieros en sistemas embebidos y para desarrolladores de *software* que se dedican al desarrollo de aplicaciones para productos embebidos.

PlatformIO es una herramienta imprescindible para los ingenieros de sistemas integrados profesionales que desarrollan soluciones en más de una plataforma específica. Además, al tener una arquitectura descentralizada, **PlatformIO** ofrece tanto a los desarrolladores nuevos como a los más avanzados una vía rápida de integración para desarrollar productos listos para su comercialización. Puede ser ejecutado en cualquier sistema operativo tal como macOS, Windows, Linux, FreeBSD.

Esta herramienta de desarrollo aplica la última tecnología de *software* escalable y flexible al mercado de los dispositivos integrados, un área tradicionalmente atendida por complejas herramientas de *software* que los experimentados ingenieros de *hardware* han aprendido con el tiempo (a menudo de forma dolorosa). En cambio, con **PlatformIO**, los usuarios pueden ser aficionados o profesionales. Pueden importar el clásico boceto "*Blink*" de **Arduino** o desarrollar un sofisticado programa **C** embebido de bajo nivel para un producto comercial. El código de ejemplo para cualquier marco de trabajo soportado puede ser compilado y subido a una plataforma de destino en minutos.

Github

GitHub

GitHub, Inc. es una corporación multinacional americana que proporciona alojamiento para el desarrollo de *software* y control de versiones usando **Git**. Ofrece la funcionalidad de control de versiones distribuidas y gestión de código fuente (SCM) de **Git**, además de sus propias características. Proporciona control de acceso y varias características de colaboración como seguimiento de errores, solicitudes de características, gestión de tareas y wikis para cada proyecto. Con sede en California, es una subsidiaria de **Microsoft** desde 2018.

Para poder entender exactamente qué es **GitHub**, primero se necesita conocer los dos principios que lo conectan:

1. Control de versiones: El control de versiones ayuda a los desarrolladores a llevar un registro y administrar cualquier cambio en el código del proyecto de sofocare a través de una bifurcación y una posterior fusión. Con la bifurcación, el desarrollador duplica parte del código fuente (llamado repositorio) para que, de forma segura, se puedan hacer cambios a esa parte del código sin afectar al resto del proyecto. Luego, una vez que el desarrollador logre que su parte del código funcione de la forma apropiada, puede fusionar este código al código fuente principal para hacerlo oficial. Todos estos cambios luego son registrados y pueden ser revertidos en caso de ser necesario.
2. Git: Es un sistema de control de versiones distribuida, lo que quiere decir que la base del código entero y su historial se encuentran disponibles abiertamente, lo cual permite un fácil acceso a las bifurcaciones y fusiones que se hayan generado con anterioridad.

Trello

Trello es una herramienta colaborativa que permite llevar la administración y gestión de proyectos, simulando un tablero *post-its*, donde se pueden obtener de forma fácil y rápida las tareas asignadas a cada uno, el estado en el que están, qué está realizando el resto del equipo y en qué parte del proceso completo está cada integrante. **Trello** se basa en la metodología **Kanban**, una metodología desarrollada por **Toyota** a principios de los años 40, dentro del sistema de gestión de la producción JIT (*Just in time*). **Kanban** no es un sistema de control o gestión, es un sistema que brinca una ayuda y mayor organización para determinar lo que se tiene que hacer en determinado momento.

Trello es un tablero que está distribuido por columnas, llamadas listas. Cada lista se compone de tarjetas, que representan las tareas a realizar, o que ya fueron realizadas. Cada tarjeta puede ser vista como una unidad básica de una lista y debe representar una tarea puntual e independiente del proyecto real, como por ejemplo, una nueva característica, un error por corregir, investigación de algún articulo, preparar un contrato, etc. La dinámica del tablero es in avanzando las tarjetas a través de las listas.

A su vez, **Trello** permite integrar otras aplicaciones para brincar más información a cada tarjeta. Entre estas aplicaciones se puede nombrar **GitHub**, que permite relacionar la tarjeta con un commit en un proyecto, adjuntar archivos a partir de **Google Drive**, checklist para comprobar el proceso de cumplimiento de la propia tarjeta, entre muchas opciones.

Thinger.io

Thinger.io es una plataforma de **IoT** de código abierto basada en la nube, desarrollada por Internet of **Thinger SL**, una empresa española cuyo objetivo es proporcionar tecnología IoT eficiente, consistente y fácil de usar.

"¿Por qué tengo que seleccionar un hardware de un proveedor compatible? ¿Por qué tengo que usar un sistema operativo sólo para mi tostadora? ¿Por qué mi dispositivo debe ejecutar un lenguaje de programación que quema las baterías? ¿Por qué tengo que depender de un proveedor específico y su plataforma cerrada?" Estas son algunas de las preguntas que se hizo Alvaro Luis Bustamante antes de empezar a trabajar en la plataforma **Thinger.io**.

Thinger.io es una nueva plataforma que está recibiendo mucho interés por parte de la comunidad tecnológica e incluso en el campo de la educación. Esta plataforma ofrece un servicio de nube listo para ser utilizado y conectar dispositivos a internet para que puedan realizar cualquier tipo de detección o actuación a distancia a través de internet. **Thinger.io** es agnóstico al hardware, por lo que es posible conectar cualquier dispositivo con conexión a internet, desde dispositivos **Arduino**, **Raspberry Pi**, dispositivos **Sigfox**, entre otros. La plataforma ofrece algunas características listas para usar, como el registro de dispositivos, comunicación bidireccional en tiempo real, tanto para la detección como para la actuación, almacenamiento de datos y configuración, de modo que es posible almacenar datos de series temporales, gestionar identidades y accesos para permitir que entidades de terceros accedan a la plataforma y a los recursos de los dispositivos a través de las API. También proporciona una interfaz web para gestionar todos los recursos y generar cuadros de mando para la vigilancia a distancia.

La principal ventaja de esta plataforma, además de ser de código abierto, es la posibilidad de obtener una comunicación bidireccional con los dispositivos, en tiempo real, mediante el uso de interfaces REST-API estándares. De este modo, es posible desarrollar cualquier aplicación de fusión de datos, es decir, de escritorio, móvil o webservice que interactúe con los dispositivos utilizando una interfaz conocida y probada basada en **REST**, mientras que los dispositivos pueden utilizar protocolos binarios más eficientes para comunicarse con la nube. Además, esta plataforma proporciona bibliotecas de clientes para conectar varios dispositivos **IoT** de última generación, como los basados en **Arduino**, **ESP8266**, **ESP32**, entre otros.

Propuesta de la solución

Dado al gran número de dispositivos de **internet de las cosas** que existen en la actualidad por persona y a su predecible aumento en los próximos años, como así también debido a que en el mercado actual casi no existen productos que sumen las habilidades de las **redes mesh** a los dispositivos **IoT** es que se ve en la necesidad de crear un producto el cual pueda ocupar ese espacio vacío del mercado.

Otro punto a valorar es que, como se vio anteriormente, no existen muchos productos que utilicen redes mesh para internet de las cosas y su precio es muy elevado como así su practicidad ya que cada modulo tiene un único objetivo de uso, esto limita el producto y no aprovecha en su totalidad las ventajas que dan este tipo de redes. Es por eso que la finalidad de este trabajo final.

Se basa en el diseño y fabricación de módulos que podrán generar una **red mesh** para ser usados en **IoT**. A su vez, cada nodo está dotado de un módulo WiFi, el responsable de la conexión y creación de la red, sensores, tanto de humedad, temperatura como aceleración y giro en los 3 ejes. Por otro lado cada nodo cuenta con salidas y entradas digitales y un relé integrado para la conexión de dispositivos de más potencia. Debido a la reutilización que se le dio a cada periférico del microcontrolador es que cada módulo cuenta con protocolos de comunicación tales como, UART, I2C y SPI, como así también una entrada analógica. Todo esto quiere decir que la red obtenida es una red completamente funcional que permite al usuario obtener datos y controlar salidas de cada módulo independientemente sin renunciar a ninguna de las funcionalidades ni características que poseen las topologías de **redes mesh**. El control de la red tendrá lugar en **Thinger.io** por medio de APIs que permiten la observación de los datos como así también el control de los periféricos de salida.

Producto	Cantidad por placa	Precio de compra	Cantidad de compra	Precio por unidad
PCB	1	$20	30	$0.67
ESP8266 - ESP 12F	1	$1.3	1	$1.30
GY 521	1	$0.67	1	$0.67
AMS1117	1	$0.29	1	$0.29
LED SMD	1	$0.03	1	$0.03
R 1.5kΩ	1	$0.008	1	$0.01
R 470Ω	1	$0.008	1	$0.01
R 10kΩ	4	$0.008	1	$0.01
C 100nF	4	$0.003	1	$0.00
Bornera 3x1	1	$0.07	1	$0.07
Bornera 2x1	1	$0.25	1	$0.25
1N4148	1	$0.05	1	$0.05
Pulsador SMD	2	$0.02	1	$0.02
Switch SMD	1	$0.06	1	$0.06
2N3904	1	$26.8	15000	$0.00
DHT 11	1	$0.78	1	$0.78
Relé 5V	1	$0.14	1	$0.14
Pines 1x20	1	$2.86	20	$0.14
SK6812	4	$0.05	1	$0.05
Carcasa	1	$1.5	1	$1.50
Fuente de alimentación	1	$1.8	1	$1.80
TOTAL:				$7.85

Tabla Nº2: Tabla de precios

Como se puede apreciar en la **Tabla Nº2** el precio total aproximado que se tendría por cada módulo en *hardware* es de US$ 7,85 y a cambio se obtiene un módulo capaz de tomar valores de temperatura, humedad, aceleración y giro como así también actuar sobre salidas digitales, salidas de potencia (relé), mostrar estados luminosos y tener pines de

comunicación. A esto se le suma las mismas capacidades que tiene una red mesh, lo que quiere decir que cada módulo sirve como extensión de la red.

Otra punto favorable que se puede encontrar en la arquitectura de este producto es el control de la red por medio de un acceso a internet lo que extiende las fronteras de uso más allá de la red local de WiFi a la cual se encuentra conectado el hub. Esto permite monitorear y controlar los módulos de la red mesh sin importar en donde se encuentre el usuario, por medio de la página web de **Thinger.io** o por su aplicación para dispositivos Android. También existe la posibilidad de implementar un control con IFTTT (*If This Then That*), esto lleva los horizontes del proyecto mucho más lejos permitiendo la incorporación de servicios como Google Assistant, Amazon Alexa, entre muchos otros.

Imagen Nº31: IFTTT

ALCANCE FUNCIONAL

Con el diseño y fabricación de este producto se busca tener un producto de bajo costo con el que se pueda automatizar procesos y obtener datos según la preferencia del usuario en lugares donde el acceso a internet no es muy fácil. Esto es por ejemplo, casa es donde la señal de WiFi no llega a la totalidad del terreno, campos en donde solo se tiene acceso a internet dentro de la casa y no fuera o ya sea un simple hogar común con acceso a una red de internet.

El alcance del producto no engloba simplemente a los hogares o al uso doméstico sino que también puede ser usado en el ámbito rural, empresarial e incluso en una ciudad inteligente. Todo caso en el que se necesite tomar datos y/o poder modificar el estado de un actuador de forma remota es un caso de uso de este producto. También puede ser usado cuando la variable a medir se encuentra sobre algún objeto no fijo, ya sea un animal o alguna especie de robot, en este caso se tiene que dotar al módulo de una batería para poder funcionar.

Las ventajas que conlleva el tener una **red mesh** para internet de las cosas es que la mayoría de las plataformas de **IoT** como **Thinger.io**, **Adafruit**, **Kaa**, **ThingsBoard** solo por nombrar algunas tienen límites de dispositivos en sus versiones gratuitas por lo que tener muchos dispositivos encestados a su plataforma no es algo posible si se quiere un bajo costo. Esta propuesta garantiza tener toda una red con un número muy elevado de nodos pero que para el servidor de la plataforma es uno solo, el hub.

Otra gran ventaja esta vinculada a las propias características de las **redes mesh** como son la auto-curación, auto-configuración, estabilidad y fiabilidad ya que pueden agregarse nodo sin modificar el funcionamiento de la red, por lo contrario agregar nodos mejora la red. En el caso de que algún modulo deje de funcionar ya sea por desconexión, rotura o falta de batería la red sigue funcionando ya que los paquetes de datos buscarán otro camino para llegar a su destinatario. La estabilidad esta dada por las dos características vistas anteriormente, es decir, entre más nodos se tengan, más caminos tendrá la información para moverse y, en el caso de que algún nodo se desconecte no se perderá ningún dato.

Domicilios

El uso de dispositivos **IoT** en los hogares es el caso de uso más frecuente y más vendido por los fabricantes ya que no a todos les interesa que la empresa en la que trabajan sea una empresa inteligente con sistemas automatizados pero, cuando se habla de volver su casa inteligente con dispositivos de internet de las cosas es cuando todo cobra sentido.

En la actualidad cada dispositivo que existe tiene una conexión directa con la red de internet, esta solución es aceptable y funcional siempre y cuando el mismo se encuentre dentro de alcance del internet.

Por ejemplo, supongamos una casa con un patio extenso y al final del mismo una piscina la cual consta de luces para su respectiva iluminación y un sistema de bombas para mover el agua. El router WiFi se encuentra dentro de la casa como es de esperar y la señal no alcanza el final del patio, volviendo inservibles aquellos dispositivos que se conecten directamente a internet para controlar el sistema de luces y de bombas y no estén al alcance.

Imagen Nº32: IoT convencional para domicilios

Como podemos apreciar en la **Imagen Nº32** un dispositivo **IoT** convencional se vuelve obsoleto al momento que no tiene alcance con la red WiFi, limitando así los lugares en los que el mismo puede ser ubicado. En la actualidad la mayorías de los hogares que tienen problemas con la señal de internet instalan extensores de WiFi, pero estos están pensados para llevar señal a esos lugares del hogar en donde la intensidad es baja o nula, no es normal que se coloque extensores para llegar a zonas como el patio y, si este fuese el caso, sería una solución muy poco económica para poder hacer uso de dispositivos de internet de las cosas.

Por otro lado, las **redes mesh** son una solución muy practica para esta problemática ya que cada módulo sirve de extensor de señal para el resto de nodos, logrando así llevar acceso a internet a rincones en donde antes era imposible llegar. Las buenas noticias no terminan acá ya que a su vez cada nodo puede ser usado para la toma de datos y el accionamiento de algo en particular, como por ejemplo el control de una luz. Véase el caso anterior, se quiere controlar el sistema de luces y bombas con el cual está dotado la pileta. Colocando un nodo intermedio se logra tener acceso a la red y este puente puede ser utilizado también para controlar una luz que se encuentra en el medio del patio. Esto mismo se puede apreciar en la **Imagen Nº33.**

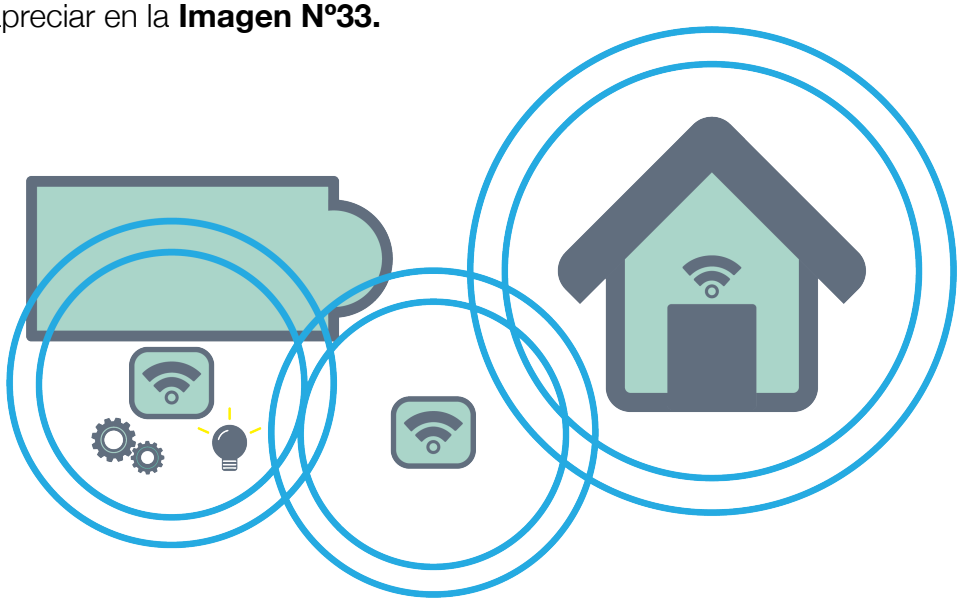


Imagen Nº33: IoT mesh para domicilios

Empresas

En el ámbito empresarial los distintivos **IoT** se están volviendo cada vez más comunes debido a que facilitan una enorme cantidad de información sin la necesidad de ningún operario que lo esté controlando. Véase así el caso de una máquina que necesita un control de temperatura cierta cantidad de veces al día por cuestiones de mantenimiento y vida útil.

Tener que mandar a un operario a realizar esta medición implica un costo adicional grande y el operario corre el riesgo de algún tipo de accidente al maniobrar la máquina. Este problema se ve solucionado mediante la integración de dispositivos inteligentes, pero no en todos los casos la señal de internet se ve presenta en las salas donde se necesita tomar las mediciones, es por esto que las **redes mesh** aparecen para solucionar este tipo de problemas.

Otro caso que se puede dar el simultáneo con el mantenimiento de ciertas maquinarias es, por ejemplo, el control de temperatura, humedad, sistemas de iluminación, cámaras, sensores de presencia, entre muchos otros. Todo esto puede ser agregado a los nodos de la red y volver una oficina simple, es una empresa inteligente que cuida la salud y bienestar de sus empleados. Cabe aclarar que todo el control de la red se puede hacer de forma remota desde cualquier lugar del mundo en donde se tenga acceso a internet, tal cual lo muestra la **Imagen Nº34**.

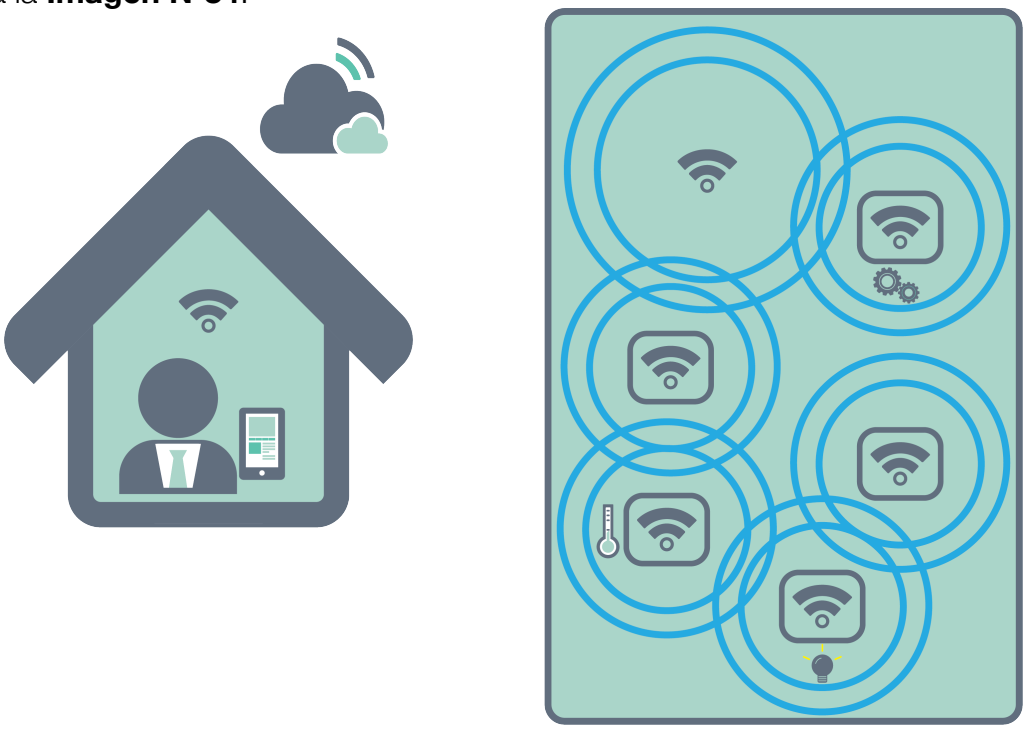


Imagen Nº34: IoT mesh para empresas

Como podemos apreciar en la imagen **Imagen Nº34** la persona encargada de la empresa o del control de la misma puede obtener los datos en tiempo real de cada uno de los nodos sin importar donde se encuentre. Ya sea para mantenimiento de cierta maquinaria, apagado o prendido de sistemas de luces, control de cortinas y ventanas y muchos otros casos que se pueden pensar. Todo esto con solamente un módulo conectado a la red WiFi y una **red mesh** para dispositivos **IoT**.

Campo

Para el caso rural se puede querer tener una medición de ciertos parámetros de los animales o mediciones de humedad del suelo en el cual se encuentran los cultivos, incluso gracias a las **redes mesh** se puede obtener la ubicación aproximada por cuadrantes de ciertos animales.

En el caso de obtener datos de animales, simplemente basta con dotar a los módulos de una pequeña batería y un *software* de bajo consumo de energía. Como el comportamiento de los animales es incierto no se podrá saber a ciencia cierta si cada nodo estará conectado a la red constantemente pero dadas las condiciones de llanura que tienen los campos se puede lograr una mayor área de cobertura.

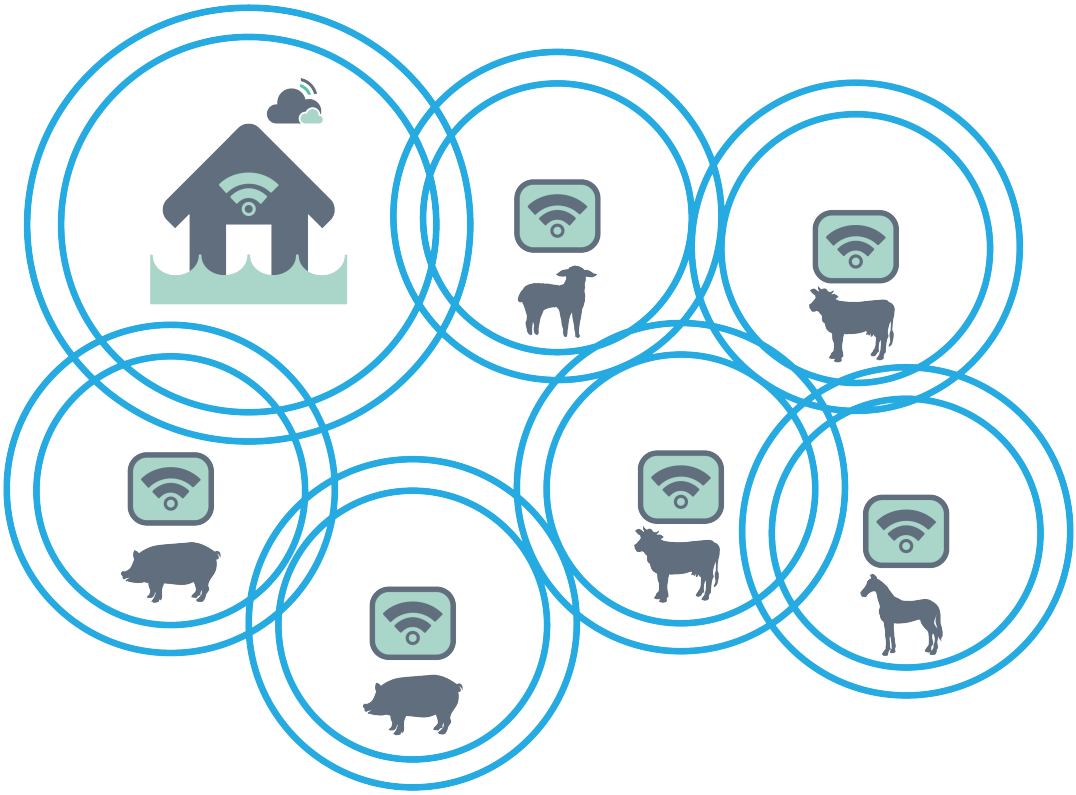


Imagen Nº35: IoT mesh para campo

Como se puede apreciar, en la **Imagen Nº35** se tiene una implementación de **red mesh** para el ejemplo de animales en un campo, Se puede ver que cada animal consta de un nodo el cual será el encargado de tomar datos de temperatura, humedad y movimiento para llevar un trackeo del mismo. Toda esta información es enviada por la **red mesh** para su posterior subida a la nube en donde, ya sea el dueño de los animales o el responsable puede tener una información en tiempo real del estado de cada uno. Esto puede ser útil

para predecir comportamiento de animales o ciertas actitudes dadas por su temperatura o sus movimientos en ciertos momentos.

Este mismo ejemplo puede ser llevado a un campo de cultivo en donde los nodos se encuentren fisicamente estáticos midiendo el nivel de temperatura y humedad de la tierra. Estos datos nos permiten tener un riego selectivo con mayor magnitud en las zonas más necesitadas y menos en las que menos lo necesitan, manteniendo un nivel estable de humedad en la tierra. También se puede pensar el caso de sensores de nivel para detectar la altura de cierto cultivo y poder predecir en que momento aproximado estarán listos para ser cosechados.

CASOS DE USO

Imagen N°36: Casos de uso

DIAGRAMAS DE ESTADO

Diagrama de entidades

Las entidades que contiene el sistema son:

- Administrador
- Invitados
- Hub
- Nodos

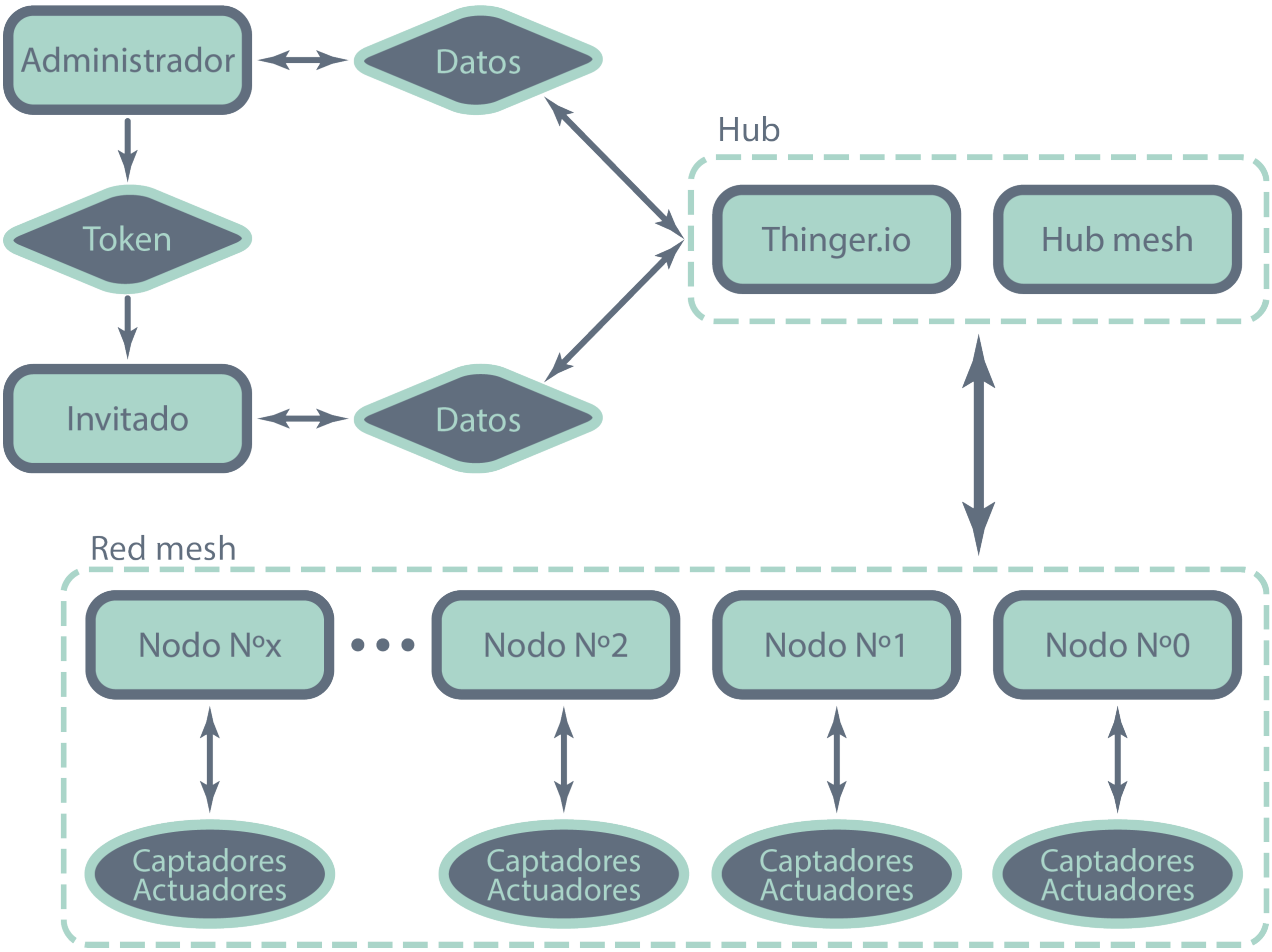


Imagen Nº37: Diagrama de entidades

En la **Imagen Nº37** se puede apreciar el diagrama de entidades en donde se logran ver claramente los agentes que forman parte del sistema. En un primer lugar se tiene el administrador, el cual es el usuario dueño de la red y quien tiene el poder de controlar la totalidad de la red como así también dar y quitar permiso a la segunda entidad de presente en el gráfico, los invitados. Estos mismos son usuarios que tienen un permiso el cual puede ser limitado o no, dependiendo del administrador, a la red para controlar los nodos y

obtener información de los mismos. Otra opción que tiene el administrador es dar permiso a ciertos usuarios por una cantidad de tiempo específica. Un ejemplo claro de uso sería un **Airbnb** en donde la persona dueña de la casa le da control de los nodos en la casa a sus huéspedes por el tiempo que dure su estadía.

Luego de esto se encuentran las entidades no humanas que son el hub y los nodos. El hub es el intermediario entre el servidor **Thinger.io** y la **red mesh** por lo que toma los datos que son entregados por las *APIs*, los interpreta y los enviar a los respectivos nodos. Caso opuesto, recibe información desde la **red mesh**, la interpreta y posteriormente la publica en **Thinger.io** para que pueda ser vista por los usuarios. Por último están los nodos, los cuales son los encargados de formar la propia red, obtener datos y actuar dependiendo de las ordenes del usuario. Cómo se puede ver en **Imagen Nº37** todas las comunicaciones dentro del sistema son bidireccionales.

Diagrama de nodos

El diagrama de nodos muestra la estructura de *firmware* con la que están programados los nodos de la red. Se puede ver que cada nodo consta de una estructura de datos llamada "*MyData*" en la cual se alojan los valores y estados de los periféricos. Para lograr esto se creo una clase de nombre "*MeshNode*" la cual sirve para representar la totalidad de la información que tiene cada nodo y poder ser enviada como una única trama de datos en contraposición con tener que enviar dato por dato. Cabe aclarar que la información dentro de este nuevo tipo de datos puede ser extensible a los periféricos que se le quieran agregar, este es un caso particular para un ejemplo de implementación en el cual podemos ver que se tiene información del ID del nodo, nombre del mismo, temperatura, humedad y aceleración en los ejes X, Y y Z. En cuanto a datos de actuación se tiene el código de control de los LED y el estado tanto del rele como de las GPIO y el pulsador.

Por otro lado se pueden observar las funciones propias de cada nodo, las cuales existen de 2 tipos dependiendo de la interacción que tengan. En un primer lugar se tiene las funciones correspondientes a la comunicación con la **red mesh** las cuales son "ReceivedCallback()" y "SendMyDataCallback()" que son las responsables de enviar, descodificar, actualizar, codificar y enviar toda la información del respectivo nodo. Caso opuesto, las funciones pertinentes a la actuación y actualización de los datos son "DHTCallback()", "AccelerometerCallback()" y "GPIOCallback()". Nuevamente cabe recalcar que estas ultimas funciones dependerán de los periféricos que estén en uso ya que se pueden agregar nuevas, quitar o modificar.

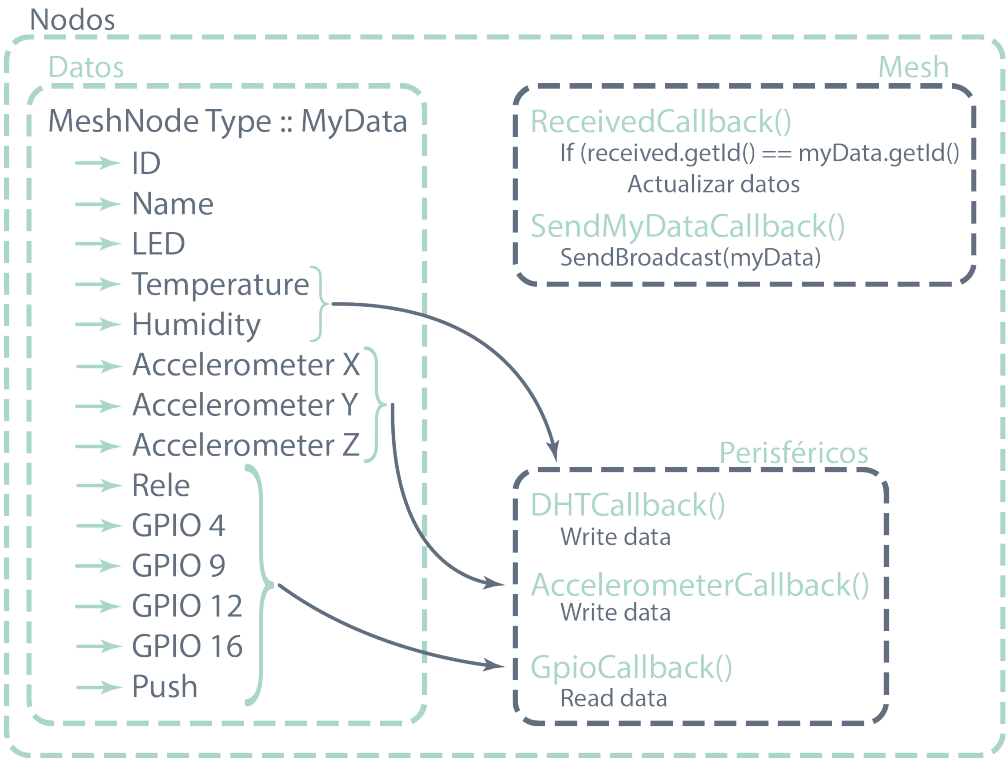


Imagen Nº38: Diagrama de nodos

Diagrama del hub

De igual manera que se vio en el caso de los nodos, se puede apreciar para el diagrama del hub. El dispositivo hub, a diferencia de los nodos no tiene una clase “MeshNode” en donde guarda sus propios datos, en este caso se crea una lista de nodos en la cual cada elemento es un “MeshNode” con la información de cada uno de los nodos de la lista.

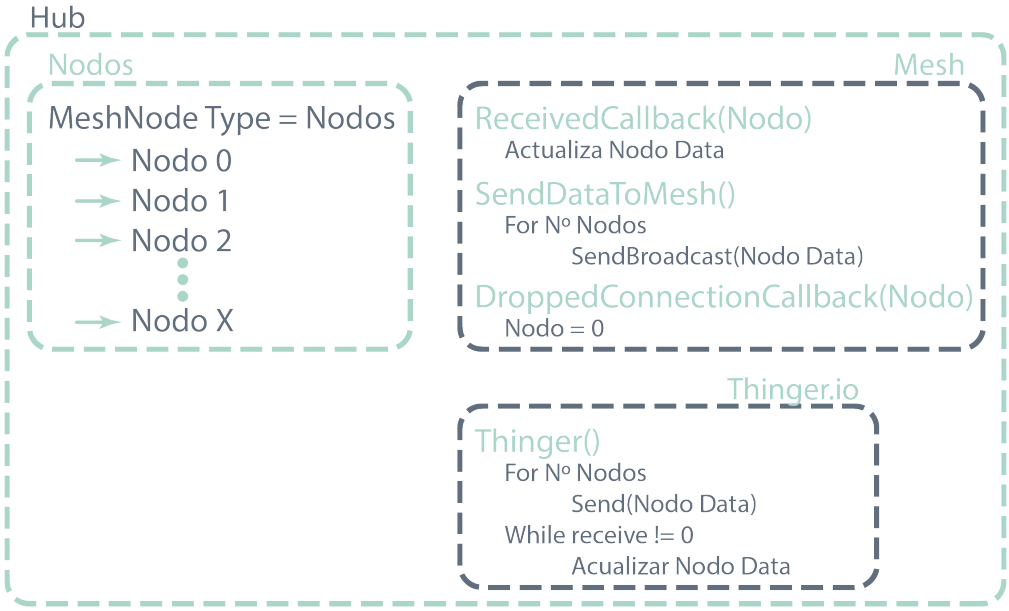


Imagen Nº39: Diagrama del hub

Como se puede apreciar en la **Imagen Nº39** la estructura básica se basa en una lista de todos los nodos que están conectados a la red y estos a su vez, con toda la información actualizada de cada uno. También se puede ver los 2 tipos de funciones que existen, por un lado se tiene las funciones que interactúan con la **red mesh** y por otro las que lo hacen con el módulo thinger.

Las comunicaciones y el cambio de información con la red se hace por medio de las funciones “ReceivedCallback()” y “ SendDataToMesh()” las cuales son las encargadas de enviar la información de acción de cada uno de los nodos hacia la red y por otro lado de recibir los datos de los nodos y actualizarlos en la propia lista. Caso opuesto se puede ver a la función “Thinger()” que se encarga de llevar a cabo una comunicación serial para la actualización de los datos de actuación de los nodos y enviar los datos tomados por los mismos hasta el servidor **Thinger.io**.

Diagrama Thinger

Por último el diagrama de thinger muestra el funcionamiento más cercano al usuario y es por medio de la plataforma antes nombrada. Al igual que la estructura de datos del módulo hub este módulo también cuenta con una lista de nodos no tiene conexión con la **red mesh**. En este caso este dispositivo sirve como un cliente para el servidor de **Thinger.io** para poder bajar datos que el usuario introduzca o también crear *APIs* para subir datos que serán vistos por el usuario.

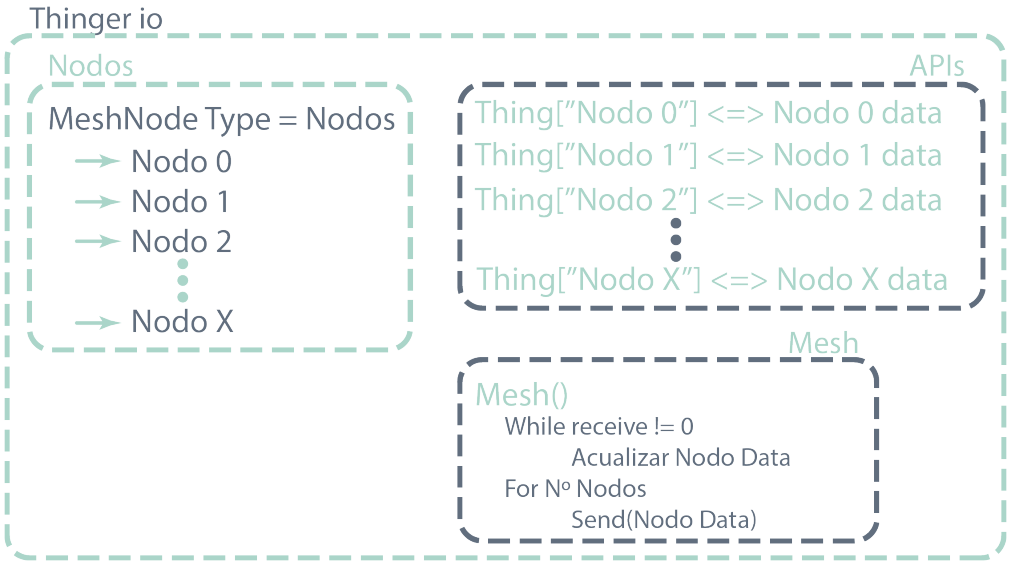


Imagen Nº40: Diagrama Thinger.io

En la se puede observar tanto la lista de nodos con toda su información como también así la creación de las *APIs* y la función encargada de la comunicación con el módulo hub.

Comenzando por la creación de las *APIs* cada nodo es propietario de su propia *API* y en ella se encuentra toda la información pertinente del mismo. Esta información se divide en entrada o salida, ya sea el caso de valores de temperatura que son mediciones que el nodo hace y envía al servidor como activación de GPIO para cambiar el estado de una salida. Para finalizar, la función “Mesh()” es la encargada de la comunicación serial con el módulo hub y en la misma se hace el envío de datos desde **Thinger.io** hacia el hub y viceversa.

Diagrama de actuación

En el diagrama de actuación se puede observar el camino que toman los datos para generar una orden de cambio de estado en un módulo en especial.

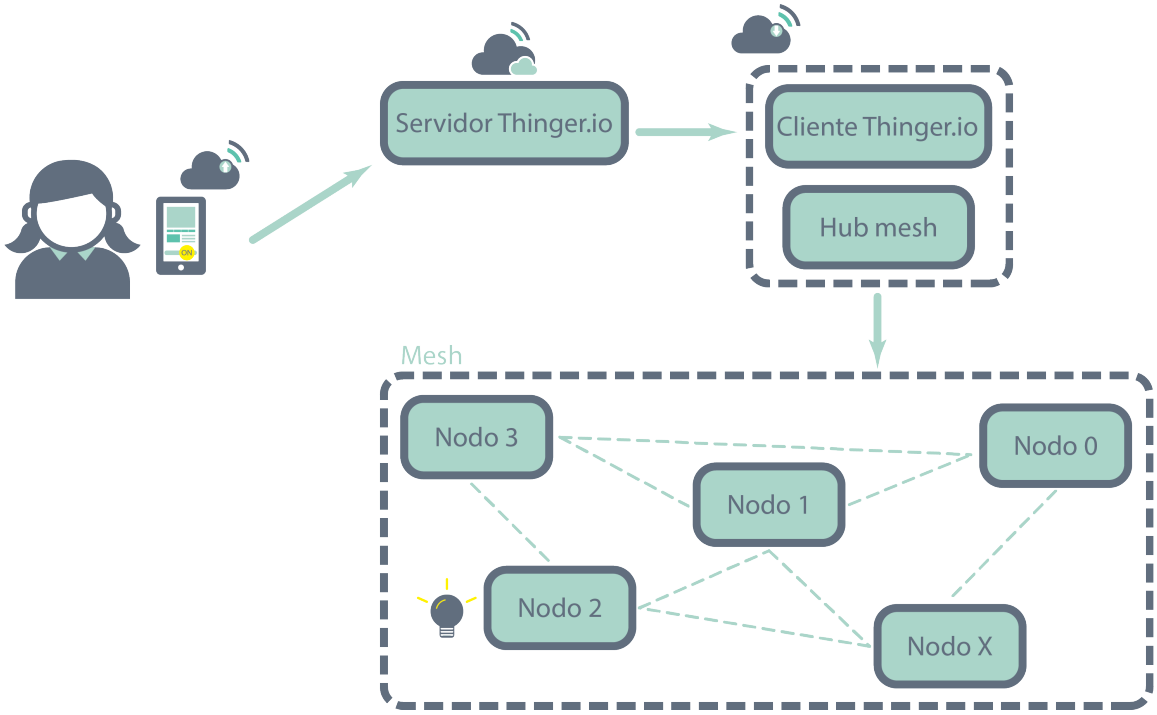


Imagen Nº41: Diagrama de actuación

Como se puede apreciar en **Imagen Nº41** la acción comienza en el dispositivo del usuario que por medio de una conexión a internet y la aplicación móvil o el navegador web da la señal del cambio de estado de una salida de cierto nodo, en el caso de este ejemplo “Nodo 2”. Esta señal viaja desde el servidor **Thinger.io** hacia el cliente ubicado en el dispositivo hub para luego ser enviar hacia la **red mesh**. El paquete de datos buscará la mejor ruta para llegar al nodo destino de la forma más eficiente y rápida, los tiempos de ruteo dependerán del algoritmo que use la red para mover los paquetes de información pero ese no es tema de análisis en este proyecto final. Una vez obtenida la señal el nodo procederá a actualizar el estado de su salida, encendiendo la luz.

Diagrama de obtención de datos

Contrario al diagrama de actuación, el diagrama de obtención de datos nos muestra el recorrido que tiene que hacer la señal para pasar de ser una medición de una variable física a ser vista en la pantalla del usuario.

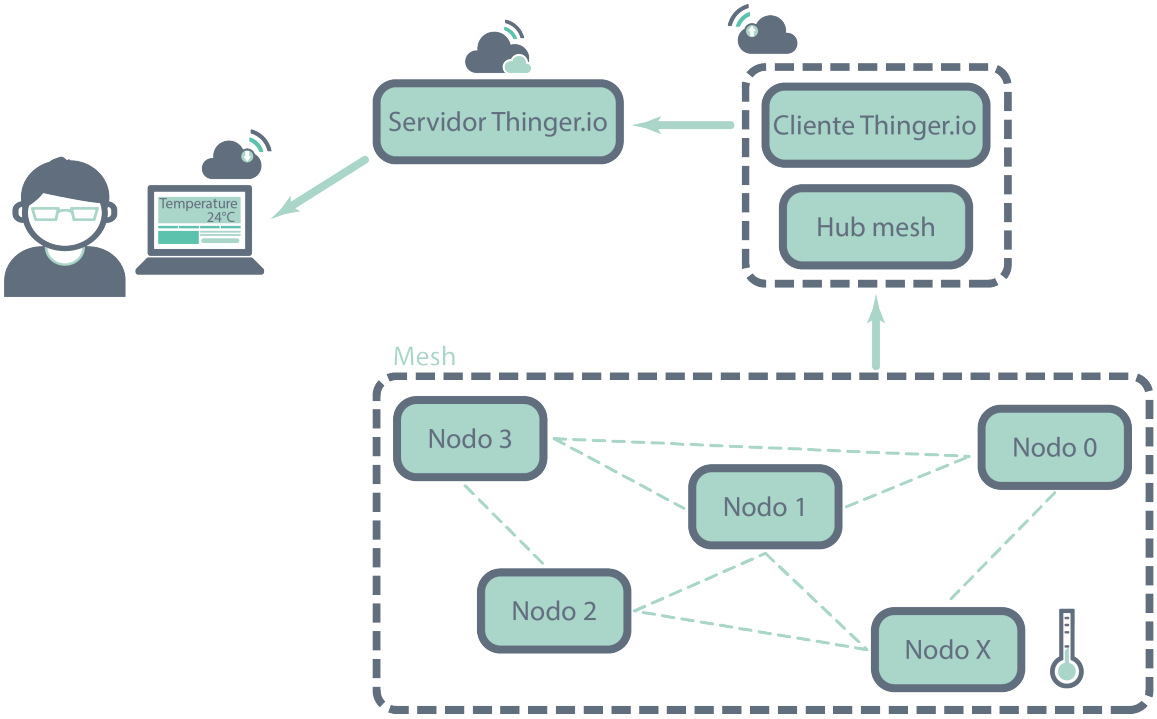


Imagen Nº41: Diagrama de actuación

La **Imagen Nº41** muestra el camino que toman los datos tomados por el nodo, comenzando por los sensores con los que cuenta el mismo (en este caso el "Nodo X" y el sensor es de temperatura). Los datos del nodo buscarán direccionarse a través de la **red mesh** hasta llegar al dispositivo hub. Una vez aquí el mismo subirá todos esta información a los servidores **Thinger.io** actualizando las *APIs*, esto permite que el usuario tenga una medición en tiempo real de la temperatura del nodo sin importar en dónde se encuentre. Nuevamente cabe aclarar que el tiempo de refresco dependerá del algoritmo de enrutado que tengo la **red mesh** como así también la velocidad de conexión con la que cuente el usuario en ese momento.

Aplicación

Luego de todo el proceso que implico llevar a cabo este proyecto final, se logro obtener una versión funcional de una **red mesh** destinada para internet de las cosas. Esta conformada por un conjunto de módulos de *hardware* y una plataforma virtual en la nube la cual sirve como interfaz de usuario.

En esta sección del documento se procederá el proceso que se siguió comenzando el diseño, pasando por la producción hasta llegar a la puesta en marcha. Se mostrarán imágenes de cada una de las partes que constituyen la total del proyecto.

DISEÑO DE LA PLACA

Partiendo de la idea final que se tiene al momento de diseñar un dispositivo **IoT** que es considerando que será algo que se encontrará en un hogar, no a simple vista pero tampoco escondido se necesita un diseño limpio, discreto y moderno. Es por eso que el tamaño y la estética importan mucho, ya que este proyecto no se basa en diseño estético de un producto se prestó más atención en el tamaño que tendría el producto final.

Es por eso que se comenzó el diseño de la placa con una idea en la cabeza, lograr un dispositivo compacto, con buenas prestaciones y estéticamente no invasivo.

Esquemático

En un principio se comenzó listando las características que se buscaba tener en cada módulo, es decir con qué *hardware* iba a estar dotado cada nodo. Una vez obtenida esta lista se paso al boceto de los circuitos destinados a cada pieza de *hardware* y control de periféricos. Se consideró que las piezas más importante con que debía contar cada módulo era:

- ESP8266 ESP-12F: El cerebro del módulo, se encarga de controlar todas los periféricos como así también las comunicaciones con la **red mesh**.

- DHT11: Un sensor de temperatura y humedad es un elemento clave cuando a internet de las cosas se refiere ya que es de mucha ayuda poder saber el valor de estas magnitudes físicas en cada rincón de la red.

- GY-521: Acelerómetro y giróscopo de 3 ejes. No hay que dejarse engañar por estas palabras, este tipo de sensor puede ser usado para detecciones de movimientos del módulo y, posteriormente, tomar acciones al respecto.

- Rele de potencia: Dado que la mayoría de electrodomésticos que se encuentran en el hogar funcionan con alta tensión, es de mucha ayuda tener un rele para poder controlar estas tensiones.

- LEDs: Este tipo de diodos se han vuelto muy famosos en los últimos años, debido a su gran capacidad emisora de luz con un bajo costo, a esto se le suma que los Neopixel son diodos emisores de luz RGB se puede generar cualquier color que se desee para indicar algo.

- GPIO: Las entradas y salidas digitales con las que cuenta el microcontrolador se encuentran disponibles para su uso por medio de una tira de pines. Esto da acceso a pines digitales con capacidad de ser configurados como PWM en cualquier momento.

- Comunicación: El ESP-12F cuenta con protocolos de comunicación tales como UART, I2C y SPI accesibles por medio de tiras de pines destinadas para tal uso.

- Pulsador: El dispositivo cuenta con un pulsador embebido en la placa que puede ser usado como entrada digital para marcar algún cambio de estado.

En la actualidad existen una infinidad de distintos tipos de sensores que podrían ser añadidos al dispositivo pero se optó por los listados anteriormente ya que son los más usados en el día a día y gracias a los pines de comunicación y GPIO se pueden adjuntan nuevos módulos.

¿Cómo se llevó a cabo el diseño y ensamblado de las placas?

Una vez obtenida la idea de los dispositivos de *hardware* que iban a ser usados, se prosiguió con llevar todo esto a la plataforma digital. En este caso se hizo uso de la herramienta de *software* libre **Kicad** ya que se considera que es una de las mejores en la actualidad y su uso es muy simple obteniendo material de calidad.

En el diseño de hardware siempre es recomendable diseñar partes por separado para que, en caso de haber errores, los mismos puedan ser detectados y reparados con mayor facilidad que si se hiciera todo junto. Esta técnica también sirve para el *testing* de cada una de las partes pero lo más importante de la segmentación de *hardware* es que en el caso de los periféricos ninguno se vuelve completamente necesario para el correcto funcionamiento del módulo. Por poner un ejemplo, si el sensor de temperatura y humedad no se encuentra en funcionamiento o simplemente el dispositivo no cuenta con uno, el mismo seguirá pudiendo conectarse a la **red mesh** y hacer uso del resto de sus perisférico con total normalidad. A continuación se detallarán los módulos esquemáticos en **Kicad**.

Módulo de alimentación:

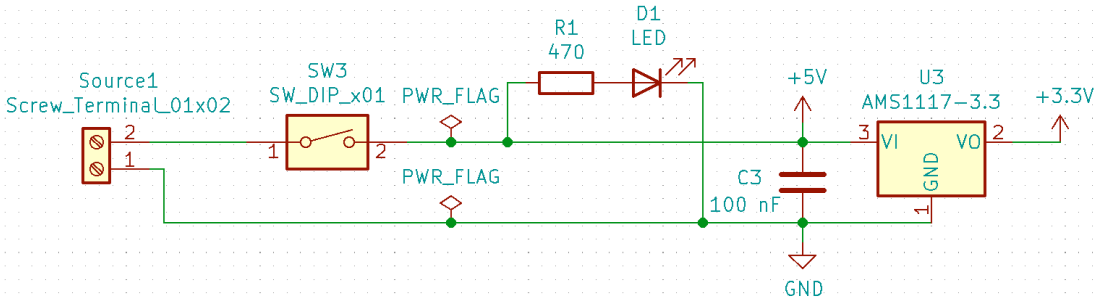


Imagen Nº42: Módulo de alimentación

La **Imagen Nº42** muestra el módulo de alimentación compuesto por una bornera de entrada y una llave para poder prender y apagar la totalidad del dispositivo. También cuenta con un led indicador que se encontrará encendido en el caso de una correcta alimentación y, de no ser así, apagado. Dicho módulo tiene integrado un regulador lineal de tensión AMS1117, el cual es el encargado de convertir una tensión de entrada Vin (4,8V - 12V) a una de salida Vout igual a 3.3V ya que es la diferencia de potencial con la que se alimenta el microcontrolador.

ESP8266 ESP-12F:

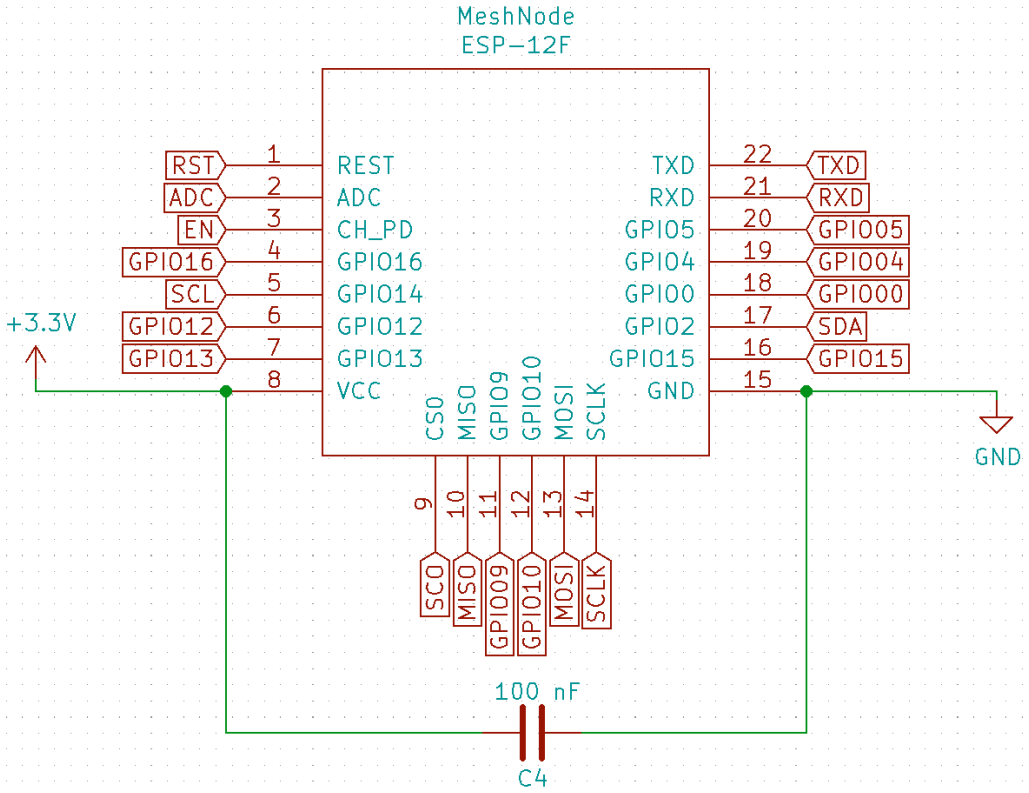


Imagen Nº43: ESP8266 ESP-12F

Cómo se mencionó anteriormente este es el módulo cerebro, el encargado de la toma de desiciones y control de todo lo que suceda en el nodo. Consta del ESP8266, un microchip WiFi de bajo costo, con un *stack* completo de TCP/IP y capacidad de microcontrolador, producido por la empresa china **Espressif Systems.** Esto nos permite tener la capacidad de conexión WiFi la cual es la tecnología usada para la creación de la **red mesh** y debido a sus características como microcontrolador es usado para el control de los periféricos adyacentes.

Módulo de programación:

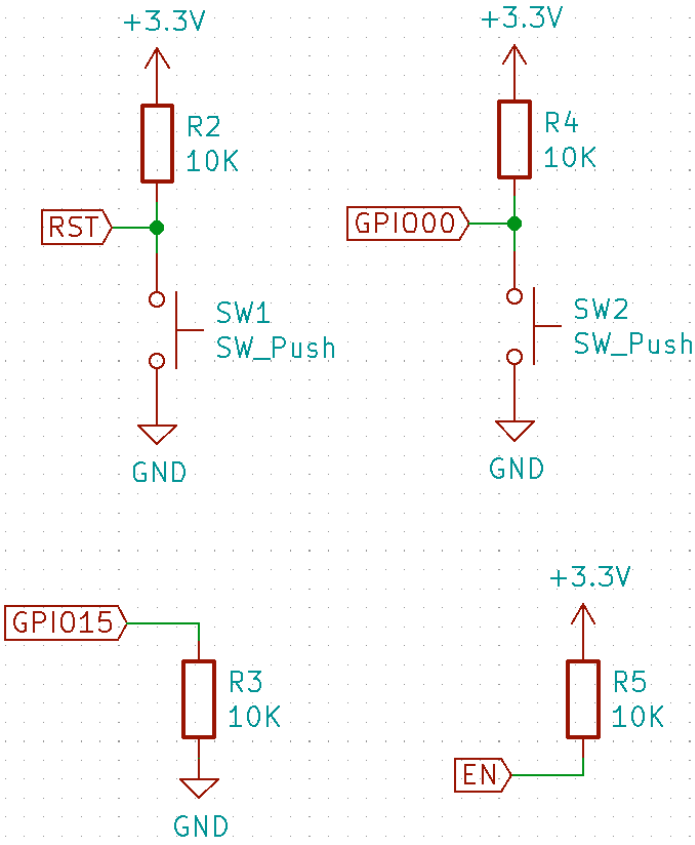


Imagen Nº44: Módulo de programación y pulsador

Imagen Nº45: Conectores de programación

Para poder poner al microcontrolador en modo de programación y que este espere el nuevo código se necesita una cierta configuración en 4 pines del mismo. Por un lado GPIO15 debe estar conectado a masa por medio de una resistencia, el pin EN contrariamente, se debe conectar a Vcc por medio de otra resistencia. En el caso de RST y GPIO00 deben contar con pulsadores y resistencias de pull-up, dichos pulsadores presionados en secuencia pondrá al microchip en modo de programación. Por otro lado el dispositivo cuenta con conectores TXD y RXD para conectar un cable USB y enviar el código desde la computadora del programador.

Acelerómetro GY-521:

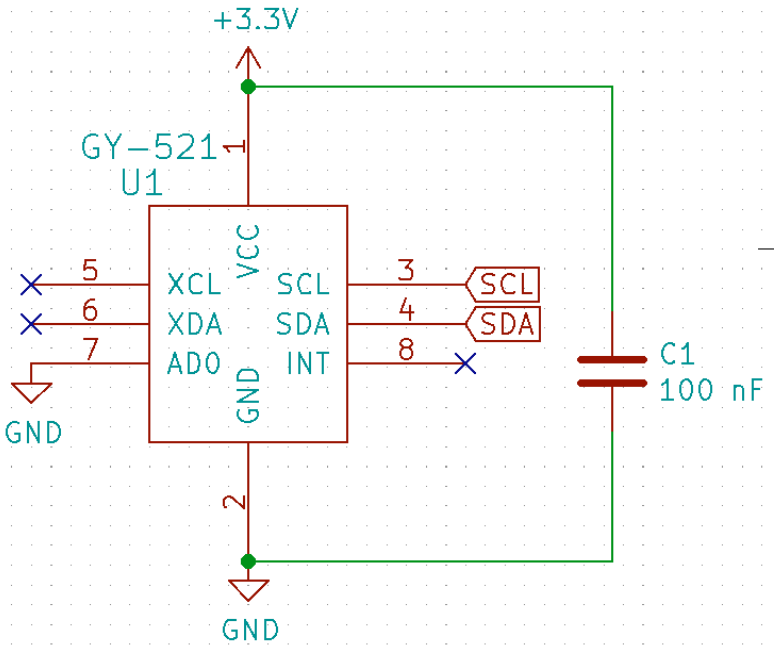


Imagen Nº46: Acelerómetro GY-521

El módulo GY-521 es un acelerómetro y giróscopo de 3 ejes (X, Y y Z) que transmite por medio del protocolo I2C. Esto quiere decir que en un principio los pines de comunicación I2C estarían destinados a este periférico pero debido a que el microchip ESP8266 no consta internamente con un módulo físico de comunicación I2C es que se por medio del uso de librerías se puede lograr este protocolo con cualquier conjunto de pines digitales.

Temperatura y humedad DHT11:

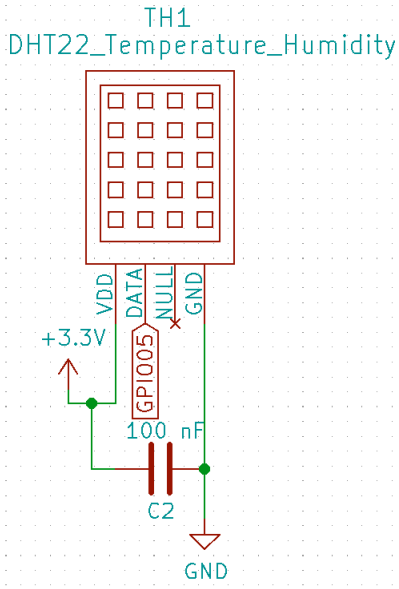


Imagen Nº47: DHT11

Como se mencionó anteriormente un sensor de temperatura y humedad es un punto clave a valorar cuando se habla de dispositivos **IoT** de uso general. En este caso el sensor DHT 11 es un sensor de bajo costo con un rango de temperatura de 0°C a 50°C y rango de humedad de 20% a 90%. Cuando se habla de sensores de temperatura precisos los precios son muy elevados, en este caso las mediciones que se harán no son de orden crítico y es por eso que el DHT11 tiene un error de ±2°C.

LEDs:

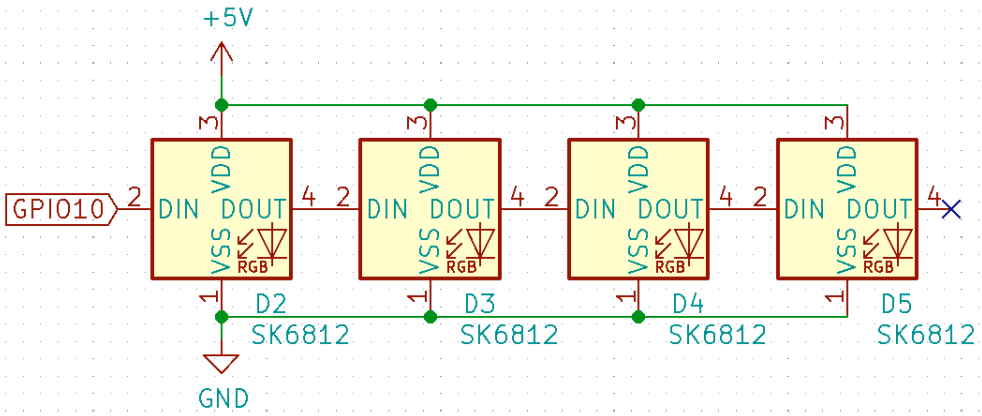


Imagen Nº48: SK6812

La **Imagen Nº48** muestra un grupo de 4 SK6812, este dispositivo es un conjunto de circuito de control y diodos emisores de luz en un único encapsulado LED. Las ventajas de este tipo de piloto luminoso en comparación al tradición LED RGB radican en la cantidad de pines con los que se controlan, en este caso se necesita un único pin digital para controlar los 4 SK6812 en cascada y poder tener la totalidad de los colores.

Conectores:

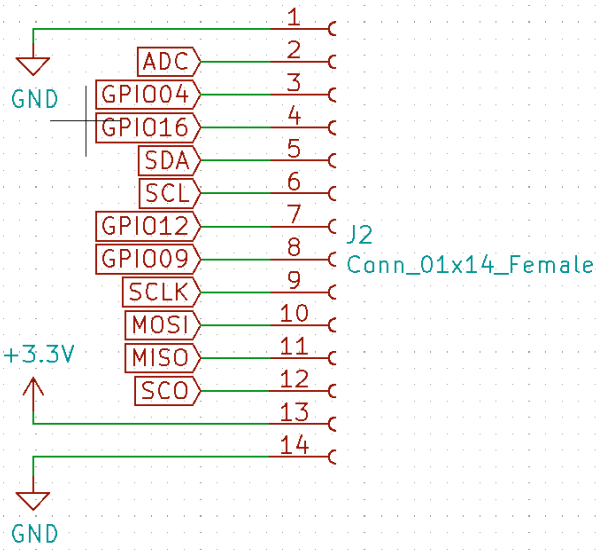


Imagen Nº49: Conectores

Este módulo se puede considerar el más simple pero a la vez es uno de los más importantes ya que cuenta con todas las entradas y salidas digitales restantes que no están en uso sumado a los pines de comunicación I2C y SPI. Como detalle no menor hay que aclarar que también se dispone de conectares a masa y a la tensión de 3.3V, todo esto es por medio de 14 pines hembra.

Rele de potencia:

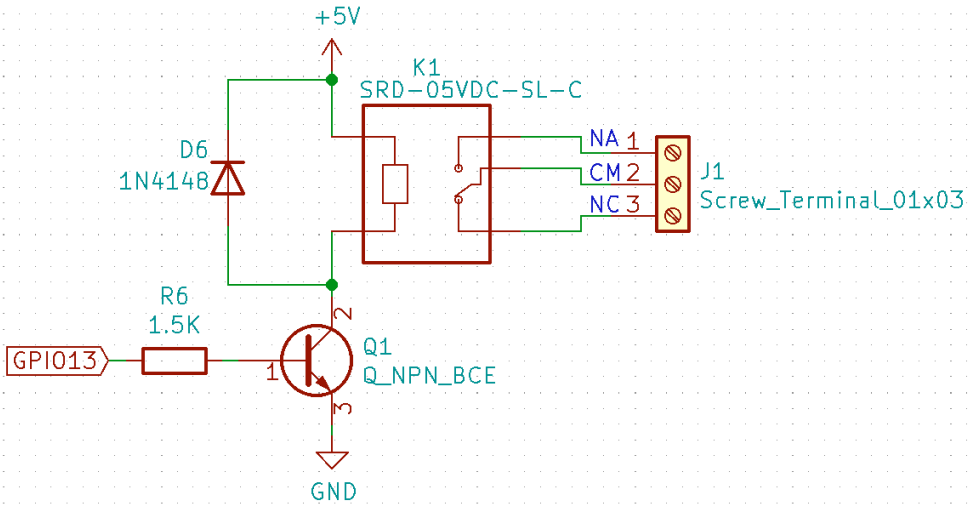


Imagen Nº50: Circuito de rele

Al igual que el sensor DHT11, este circuito es una pieza infaltable al momento de diseñar un dispositivo de internet de las cosas genérico ya que la mayoría de electrodomésticos o aparatos que se quieran controlar manejan una potencia mayor a la del propio nodo. El circuito esta formado por un transistor NPN el cual es el encargado de activar la bobina del rele, lo que cambia un cambio de estado en el mismo. La bornera de salida cuenta con 3 terminales, masa, normal abierto y normal cerrado dependiendo de cuál será el caso para el que se necesite.

PCB

De las siglas en ingles "*Printed Circuit Board*" o **placa de circuito impreso** es la encargada de soportar y conectar los componentes electrónicos, con vías o pistas de cobre, para que un circuito o dispositivo electrónico funcione como se desea.

Una vez obtenido el circuito esquemático, la misma herramienta **Kicad** permite transformar los componentes de símbolo a su tamaño y forma físico. De esta forma los mismos pueden ser ubicados la placa de la forma que el diseñador lo desee y hacer las conexiones debidas para que todo quede tal cual lo muestra el esquemático. En este caso se utilizó un circuito impreso de dos capas por lo que si bien la mayoría de los

componentes se encuentran en una única capa, la segunda también aloja algunos de ellos y sobre todo fue usada para llevas vías de conexión.

Otro punto a tener en cuenta a la hora de diseñar un circuito impreso es la disposición de los componentes ya que nada impide una libre ubicación de los mismos pero el uso y la estética de la placa son factores que pueden peligrar. Dado el ejemplo de una bornera la cual solo tiene acceso para los cables en uno de sus lados, si esta es ubicada de tal manera que no se pueda hacer correcto uso queda totalmente inutilizada y, por ende, el circuito también. Es por eso que todos los conectores se encuentran en los bordes de la placa y no en el medio.

Al estar trabajando con señales de alta frecuencia como lo es el WiFi, es necesario el uso de una antena, en este caso esta se encuentra embebida dentro de la misma placa del microcontrolador y según recomienda el fabricante (*Anexo*) la misma se debe encontrar a al menos 15mm de cada lado y 15mm al frente, como se puede apreciar en la **Imagen Nº51.**

Las dimensiones finales que se lograron son 61,75mm x 60,85mm x 2,4mm lo que hace al dispositivo compacto como se tenia planeado. A continuación se muestra el circuito **PCB** en la herramienta **Kicad** de 3 maneras distintas, en primer lugar el ruteo de la parte superior de la placa, en segundo el ruteo de la parte inferior y por ultimo de ambos lados.

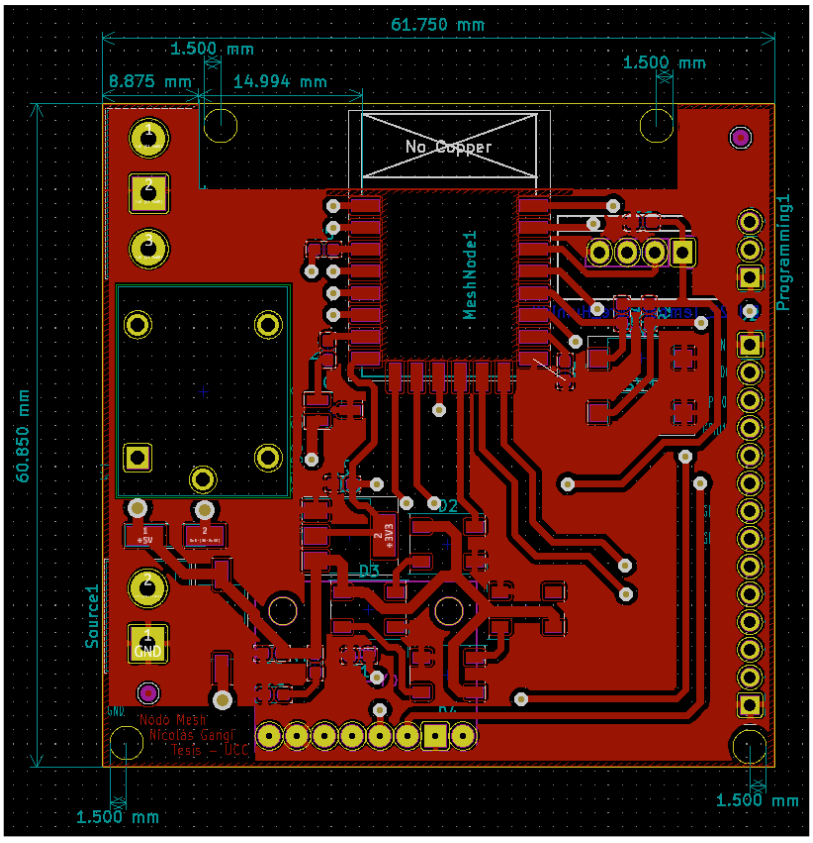


Imagen Nº51: PCB parte superior

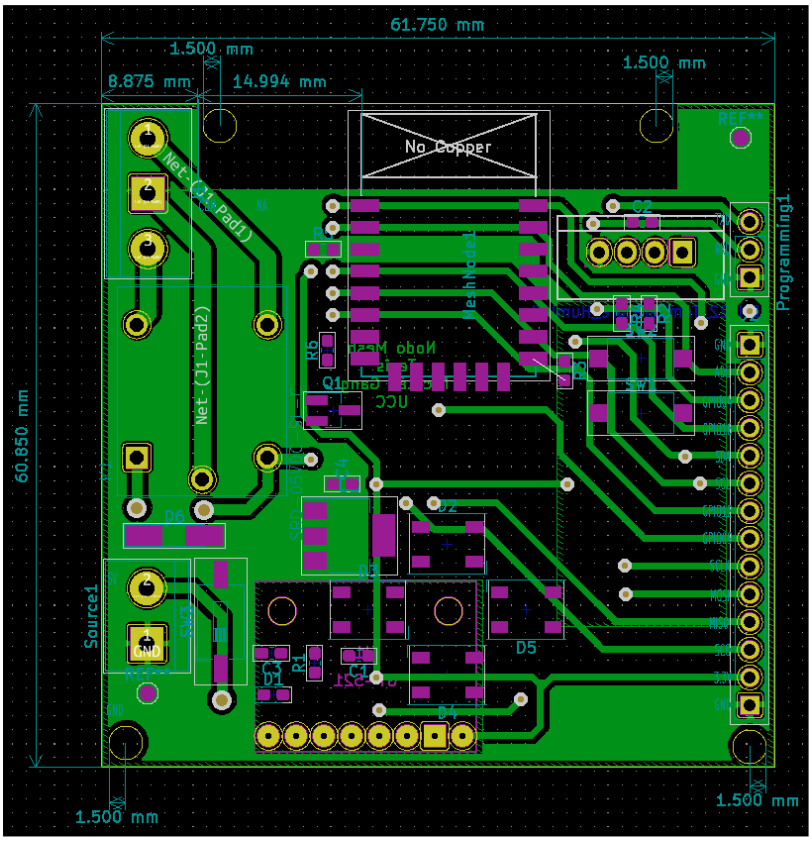


Imagen Nº52: PCB parte inferior

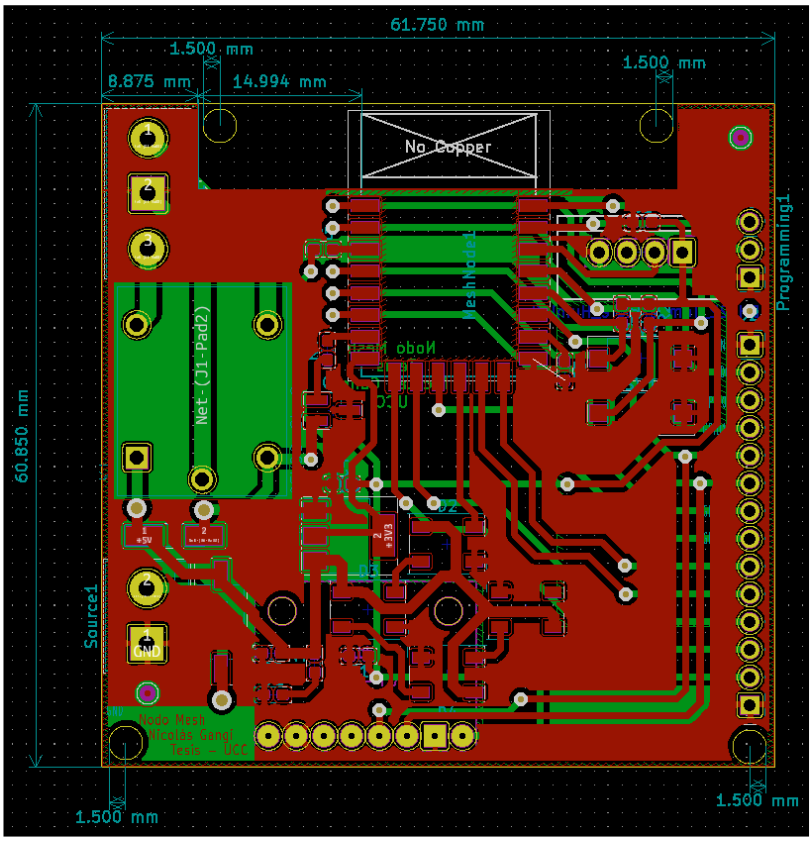


Imagen Nº53: PCB

Otra gran opción con la que cuenta la poderosa herramienta de **Kicad** es la visualización 3D ya que permite tener una noción de cómo quedará el producto final luego de colocar los respectivos componentes. Un detalle a tener en cuenta es que no todos los componentes cuentan con su modelo 3D dentro de la propia librería de **Kicad** por lo que a veces es necesario tener que descargar por cuenta propia dichos modelos para poder adjuntarlos al modelado final. Como se dijo anteriormente, es simplemente una herramienta de visualización, este proceso no va a cambiar nada a nivel de funcionamiento ni fabricación, simplemente permite una vista previa para esos casos en el cual el diseño y la estética del **PCB** es un favor importante.

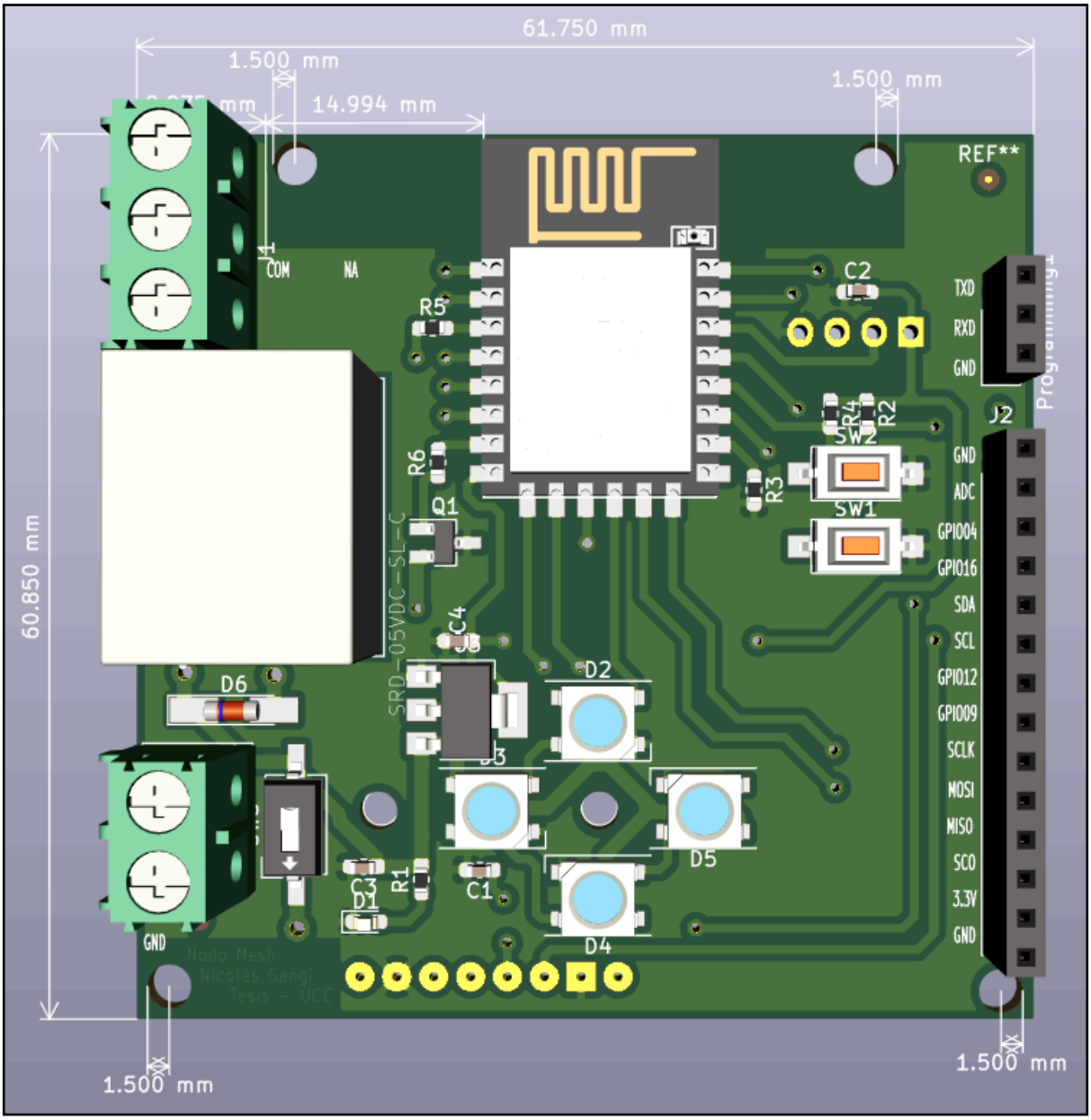


Imagen Nº54: Vista en 3D

Detalles a tener en cuenta sobre el diseño. El prototipo cuenta 4 orificios para el anclaje a una carcasa si es deseado. También tiene otros dos orificios para un anclaje más seguro del acelerómetro GY-521 aunque eh de notar que con los puntos de soldadura se nota un agarre fuerte y sólido.

Las vías destinadas al manejo de potencia del lado del rele son de un ancho mayor para soportar más corriente.

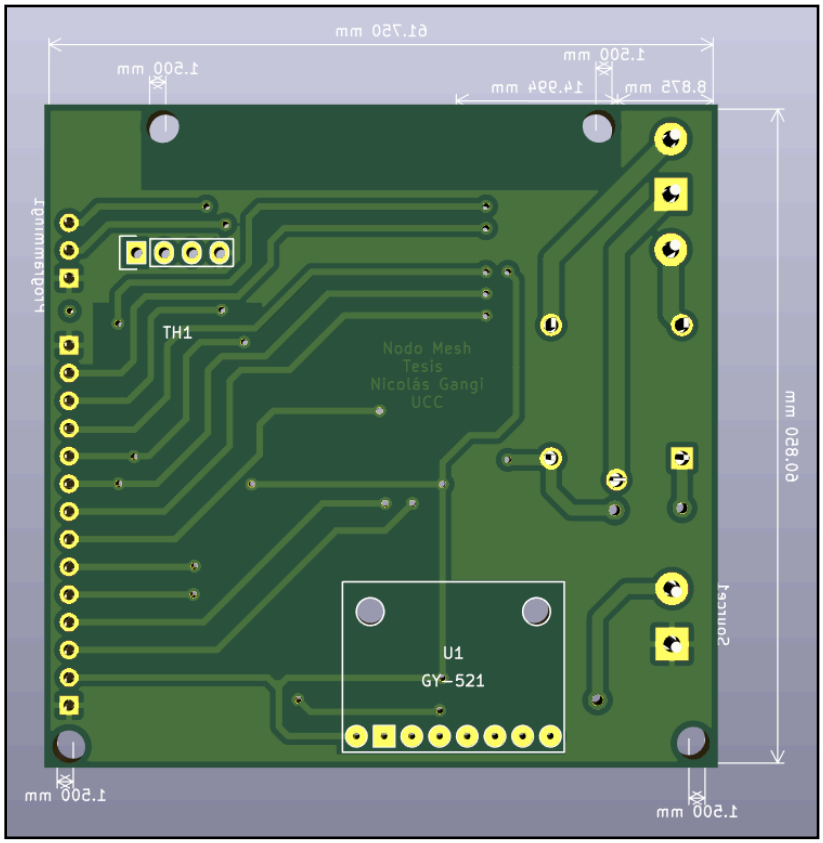


Imagen Nº55: Vista en 3D

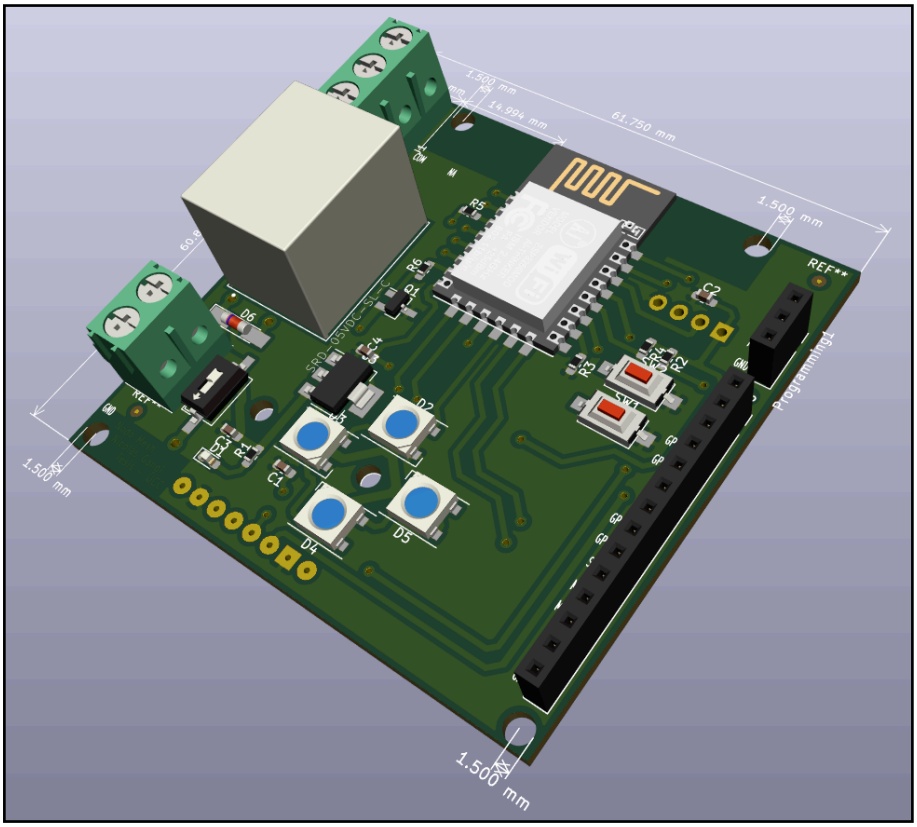


Imagen Nº56: Vista en 3D

En el caso de que el circuito tenga que ser fabricado a gran escala, en el diseño se tuvo en cuenta dispone de puntos (véase en la **Imagen Nº56** REF**) que servirán como referencia si se desea colocar los componentes por medio de una maquina *Pick & Place*, lo que no quieta que se pueda hacer de forma manual.

Producción

Una vez obtenido el diseño del **PCB** en **Kicad** se prosiguió con su producción. En este punto se tienen 2 opciones, por un lado enviar el diseño de la placa a una empresa especializada la cual se encargue de la producción o hacerlo por cuenta propia. En el caso de este proyecto se optó por enviar el diseño a una compañía china llamada **JLCPBC**. **JLCPBC** es la empresa líder mundial en la fabricación de prototipos de **PCB** y un fabricante de tecnología especializado en la producción rápida de prototipos de **PCB** y pequeños lotes de los mismos. Las ventajas que se tuvieron por haber elegido esta opción fueron la calidad del producto obtenido, fiabilidad y una terminación que solo puede ser obtenida con cierta maquinaria que solo cuentan algunas pocas empresas. Por otro lado la desventaja de esta opción fue el tiempo de demora que se tuvo, cabe aclarar que el pedido se inicio al mismo tiempo que dio comienzo la pandemia por el **Covid-19** por lo que los envíos fueron levanten retrasados, a esto sumarle que la empresa se encuentra en el país asiático de China.

Las placas obtenidas se pueden observar en la **Imagen Nº57** y la **Imagen Nº58**.

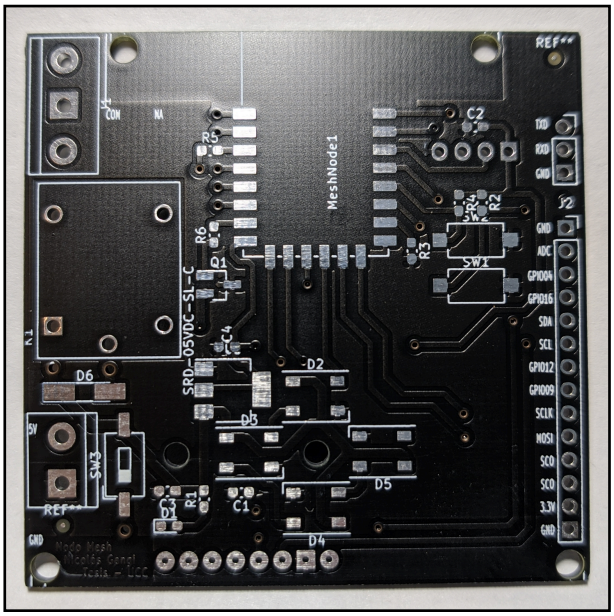


Imagen Nº57: PCB sin componentes

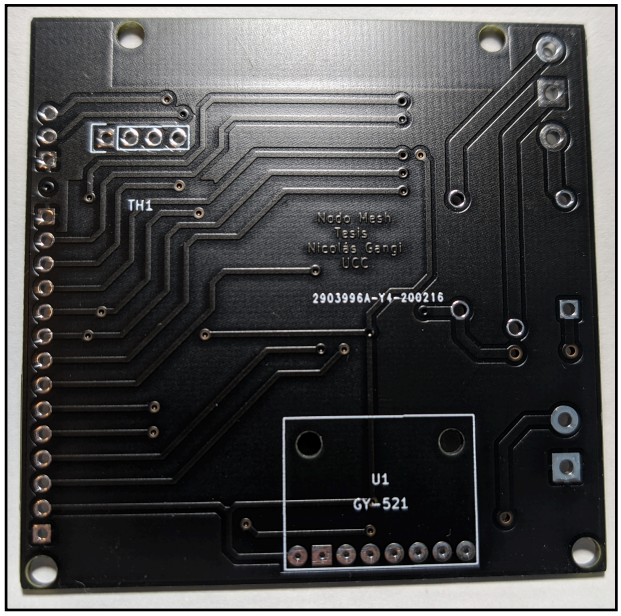


Imagen Nº57: PCB sin componentes

Se puede apreciar la calidad de construcción que se obtuvo y el precio de las mismas fue de solamente US$ 7,00 sin duda una muy buena opción a tener en cuenta a un precio más que accesible.

Como es de esperar al momento de ensamblar un primer prototipo electrónico por más buen diseño y precaución que se haya tenido siempre tiene que existir aunque sea un pequeño error, es parte del aprendizaje. En este caso se tuvieron 2 errores al momento de ensamblar los componentes, es de agradecer que fueron errores fáciles de reparar y se pudo seguir adelante con el armado de las mismas. Por un lado el diámetro de las perforaciones destinadas al sensor GY-521 fueron hechas de un ancho mejor al real por lo que antes de insertar el componente que tuvo que limar los propios pines para que el mismo pueda ingresar correctamente, un error muy pequeño pero no por eso inexistente.

Por otro lado, el *layout* del transistor del circuito del rele de potencia no era el correcto, teniendo los pines de colector y base cambiados entre si, en este caso y debido a que en el mercado local solamente se encontró un único transistor que cumpliera las especificaciones se tuvo que rotar al mismo sobre su propio eje para ser soldado al revés, de esta forma se tiene un funcionamiento correcto. En este caso el error es un poco más importante pero nuevamente pudo ser solucionado. Ambos errores fueron tenidos en cuenta y corregidos para el caso de un segundo prototipo.

Una vez obtenidas las placas lo siguiente es la compra de componentes la cual se puede ver detallada en **Tabla Nº2**. Nuevamente las compras se hicieron por medio de la pagina web **Ebay** pero gracias a las políticas locales de Argentina y la pandemia del **Covid-19** solamente se recibieron 5 de la totalidad de los componentes, es por esto que se prosiguió a la compra de los componentes faltantes en las tiendas locales por un precio mucho más elevado. Una desventajas de las tiendas locales es que no siempre cuenta con el stock de los productos que se buscan, dado el ejemplo del transistor antes mencionado. En el caso de este proyecto se pudo conseguir casi la totalidad de los componentes a excepción de 2, el *switch* SMD y los SK6812. Esto no significa un incorrecto funcionamiento de los nodos ya que se suplantó el *switch* que era destinado a prender y apagar el dispositivo por un cable, lo que trae como desventaja la incomodidad al momento de prender y apagar el mismo. Por otro lado los LED SK6812 son un tipo de piloto luminoso difícil de conseguir debido a su falta de stock a nivel nacional por lo que el prototipo actual no cuenta con los mismos, esto no entorpece el funcionamiento pero si le quita una cualidad muy importante.

Cómo se menciono anteriormente el precio final debido a la compra local de componentes fue un poco superior a lo planeada y por ende, en menor cantidad de algunos de los componentes tales como el módulo rele de potencia y DHT11.

Para el armado de las placas se tenía planeado hacerlo en las instalaciones de la **Universidad Católica de Córdoba** para hacer uso de su maquina *Pick & Place* pero, nuevamente la pandemia por el **Coronavirus** que tomo a todos por sorpresa cambió las cosas y se tuvo que pensar en nuevos planes. Se opto por un ensamblado casero con soldador de estaño y colocar los componentes a mano, como ventajas propiamente dichas esta metodología no tiene ninguna pero ya que no se tenia otra opción es con lo que se prosiguió.

En el prototipado de placas electrónicas se acostumbra, si se tiene que soldar componentes de montaje superficial (SMD), a hacerlo en un orden de tamaño de menor a mayor. Esto se debe a que una vez colocados y soldados los componentes más grandes es muy difícil poder acceder a los más pequeños. En este caso se comenzó colocando las resistencias y capacitares y se siguió adelante con reguladores, pulsadores, ESP8266, hasta llegar las borneras y el rele.

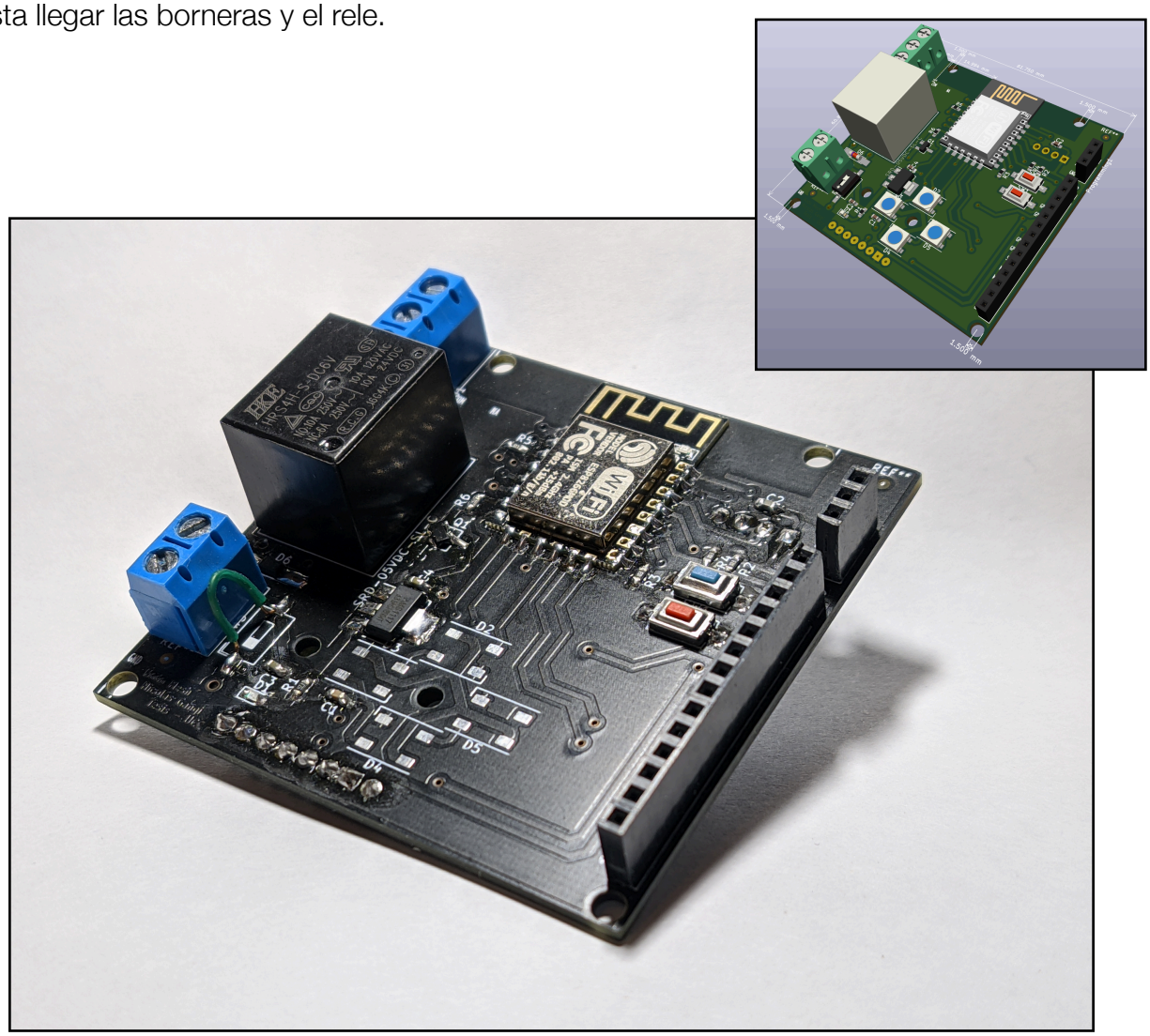


Imagen Nº58: Placa terminada

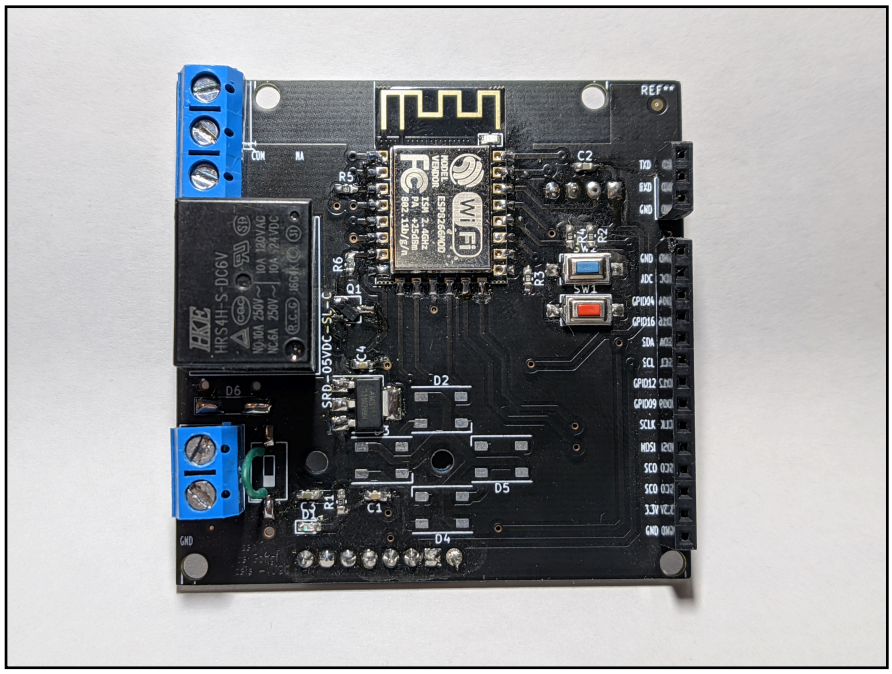


Imagen Nº59: Placa terminada

En el margen inferior izquierdo de la **Imagen Nº59** se puede apreciar uno de los problemas antes mencionados que tiene que ver con la falta del *switch* SMD. En este caso se lo suplantó con un cable lo que se podría ver cómo una llave constantemente cerrada pero sin posibilidades de ser abierta.

La **Imagen Nº60** muestra claramente la ubicación de los sensores DHT11 (temperatura y humedad) y GY-521 (aceleración y giro en 3 ejes). Su ubicación en el dorso de la placa está dada con el objetivo de poder reducir espacio en la misma y que se vuelva un dispositivo aún más compacto.

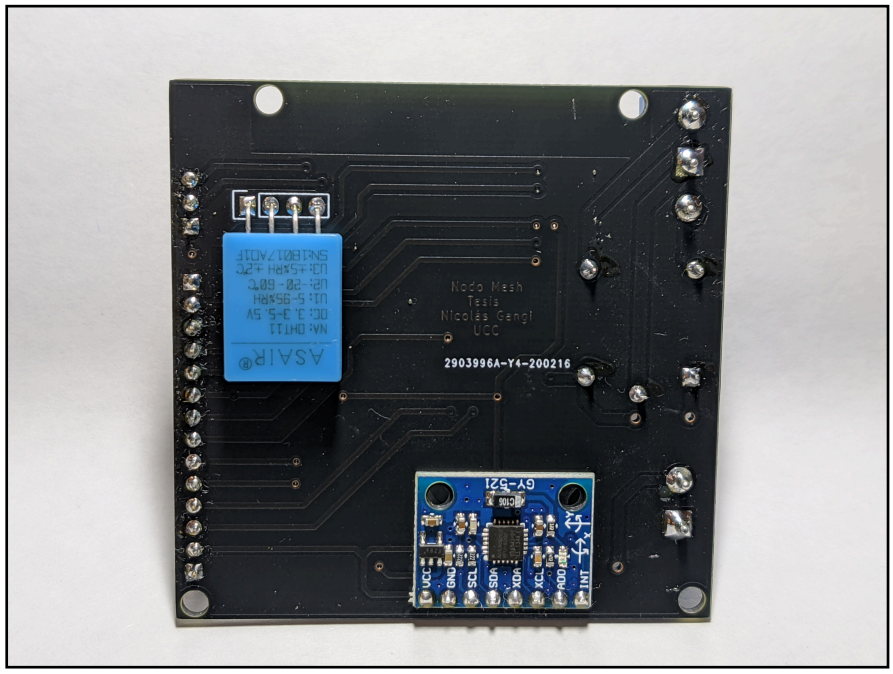


Imagen Nº60: Placa terminada

Puesta en marcha

Inmediatamente después de haber finalizado con el ensamblado de las placas se prosiguió a medir la continuidad de las mismas con el objetivo de buscar algún corto circuito o circuito abierto que se pueda haber producido producto de la soldadura. Para este proceso de hizo uso de un multímetro digital tal cual muestra la **Imagen Nº61**.

Imagen Nº61: Multímetro digital

Este tipo de instrumentos es muy usado en la rama de la ingeniería electrónica debido a su portabilidad y versatilidad, fue una pieza fundamental en el desarrollo de este trabajo. Una vez finalizada la medición de continuidad que se le hizo a cada una de las placas se prosiguió a conectarlas una por una para medir las tensiones dentro del circuito, tensiones tales como Vin, Vout de regulador, tensión de alimentación del ESP8266, tensión de alimentación de los sensores, etc.

Para la alimentación de los módulos se usaron transformadores variados debido a que en este caso no se contaba con una gran cantidad de fuentes de alimentación iguales. Recordando el rango de tensión de alimentación permita por el regulador de entrada es de 4,8V a 12V por lo que mientras que los valores de la fuente se encuentre dentro de este rango el módulo funcionará correctamente.

Como detalle es bueno aclarar que el componente AMS1117 es fácil de usar y está protegido contra cortocircuitos y sobrecargas térmicas. Los circuitos de protección térmica apagarán el regulador si la temperatura de la unión excede los 165°C en el punto sensorial. Consta de un *layout* compatibles con los antiguos reguladores ajustables de tres terminales, este dispositivos ofrece la ventaja de un menor voltaje de salida, una tolerancia de referencia más precisa y una mejor estabilidad de referencia con la temperatura.

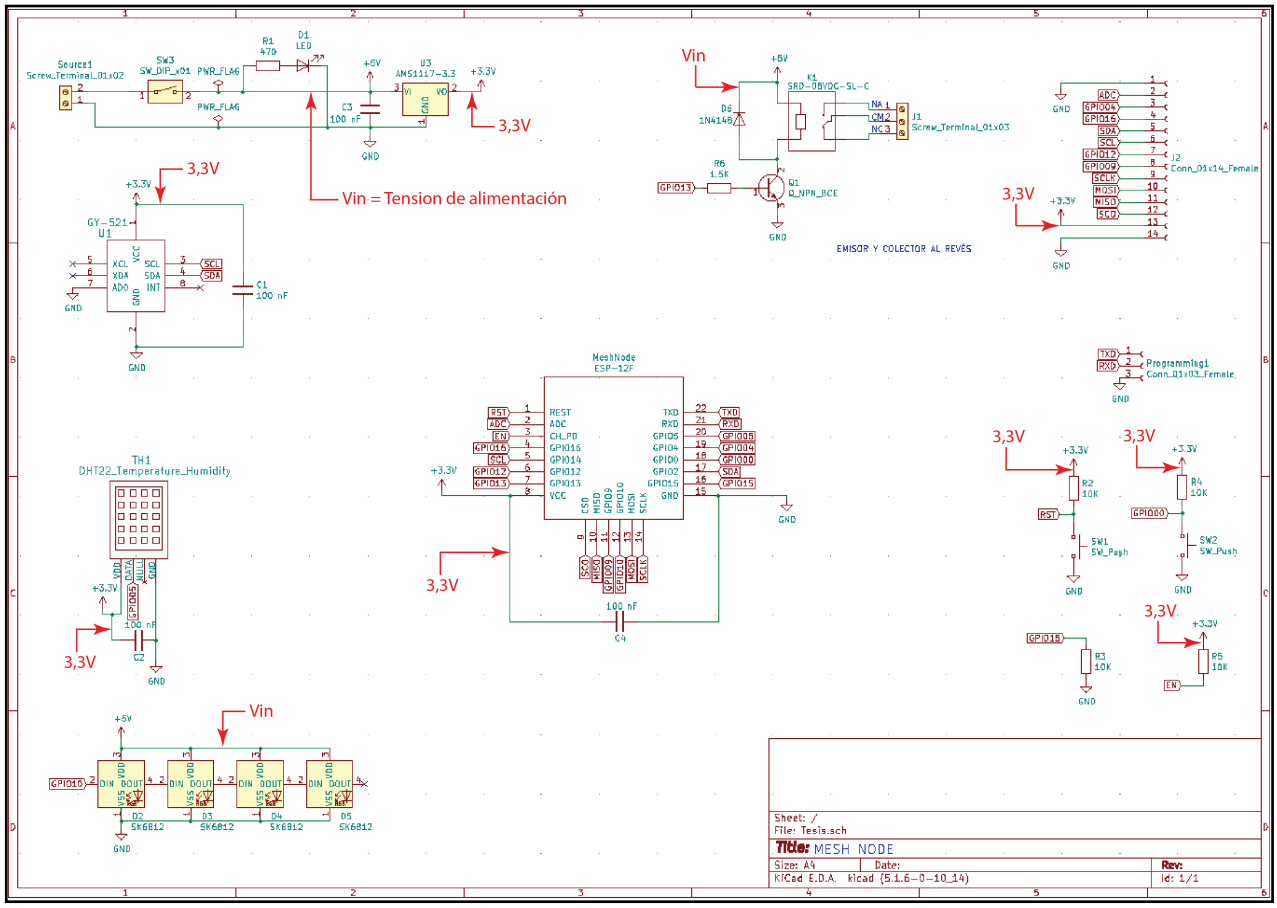


Imagen Nº62: Esquemático completo

Como se habló anteriormente y tal cual se muestra en la **Imagen Nº62** se optó por la medición de tensión en ciertos puntos del circuito con el objetivo de corroborar su correcto funcionamiento. También se puede apreciar que la mayoría de los puntos de medición son valores de 3.3V, esto se debe a que los componentes en su mayoría se encuentran alimentados con este valor de tensión. Contrario a esto se observa que ciertos componentes tales como los SK6812 y el rele de potencia van conectados directamente a la tensión de alimentación del circuito.

Un punto desfavorable que tiene el diseño de estos dispositivos radica en la falta de incorporación de un circuito conversor UART a USB embebido en la placa. Este tipo de circuito permite al programador tener acceso a programar el chip de una forma mucho más sencilla por medio de un cable USB directamente conectado a la placa. En la actualidad no todas los módulos que se comercializan en el mercado tienen incorporado este añadido de hardware y la razón se debe a una reducción de costos y espacio. Dado que no este proyecto no cuenta con ese conversor es que se creó el módulo de programación tal cual muestra la **Imagen Nº44** y **Imagen Nº45**. Esto nos permite tener acceso a los pines de

TXD y RXD a los cuales se le incorporará un módulo UART-USB externo para lograr programar el circuito. Se puede ver dicho módulo en la **Imagen Nº63**.

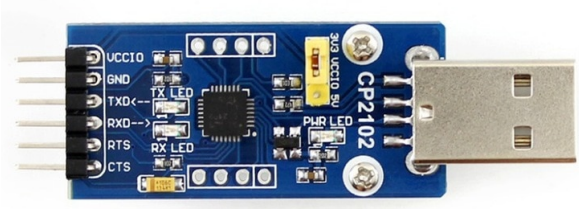


Imagen Nº63: Módulo UART a USB

Para poder ser programado el ESP8266 necesita ingresar en un modo de programación, de esto es lo que se encargan los circuitos de programación pero en el caso de no tener uno incorporado se lo puede reemplazar por medio de una simple configuración de resistencias y pulsadores, nuevamente esto ve reflejado en **Imagen Nº44** y **Imagen Nº45**.

El proceso de programación es simple y se basa en enviar el código compilado hacia el microcontrolador por medio de la herramienta de *software* elegida por el programador y luego de esto presionar el pulsador ubicado en el pin GPIO00, a continuación y sin soltar dicho pulsador se debe presionar el botón de reset RST. Lo que genera esto es que el chip entre en modo de programación y el código enviado logre ser cargado correctamente. Todo el procedimiento se fue obtenido gracias a "***Instructables circuits***" en el sitio web:

Código Nº21: Instructables circuits

Se comenzó con ejemplos simples para probar cada una de las partes del *hardware* y si funcionaba todo correctamente.

- Blink: Hacer parpadear el LED incorporado en el ESP8266 cada 1 seg.
- Hello world: Imprime por pantalla el mensaje "Hello world".
- GPIO: Cambiar de estado los GPIO y el rele de potencia.
- Blink con pulsador: Prender y apagar el LED incorporado por medio del pulsador.

- DHT11: Leer e imprimir los valores de temperatura y humedad.
- GY-521: Leer e imprimir los valores de aceleración y giro en los 3 ejes.
- Mesh: Generar una **red mesh** y conectarse a la misma.
- MeshNode: Probar la creación y funcionamiento del nuevo tipo de dato.
- ThingerDemo: Comprobar el correcto funcionamiento de la plataforma **Thinger.io**.

Estos son los principales ejemplos que se usaron para probar el correcto funcionamiento de las placa y como interactúan entre ellas. Luego de esto se prosiguió a desarrollo puro y duro de cada tipo de *software* ya que se cuenta con 3 tipos distintos: nodos, hub y thinger.

SOFTWARE NODOS

La idea general del *firmware* incorporado en cada uno de los nodos se basa en una estructura de toda su información, en ella se pueden ver los valores y estados de cada uno de sus periféricos. Cabe aclarar que la estructura diseñada es solo un ejemplo de implementación que puede ser extendido casi hacia el infinito. En este caso cada estructura esta formada por:

- ID: Número único de cada nodo.
- Nombre: Nombre único de cada nodo, desde el 0 en adelante.
- LED: Valor numérico que simboliza el color de los LED.
- Temperature: Valor de temperatura del nodo.
- Humidity: Valor de humedad del nodo.
- Accelerometer X: Valor de aceleración en el eje X del nodo.
- Accelerometer Y: Valor de aceleración en el eje Y del nodo.
- Accelerometer Z: Valor de aceleración en el eje Z del nodo.
- Rele: Estado del rele del nodo. ON/OFF.
- GPIO4: Estado del GPIO4. ON/OFF.
- GPIO9: Estado del GPIO9. ON/OFF.
- GPIO12: Estado del GPIO12. ON/OFF.

- GPIO16: Estado del GPIO16. ON/OFF.
- Push: Estado del pulsador del nodo. Presionado/suelto.

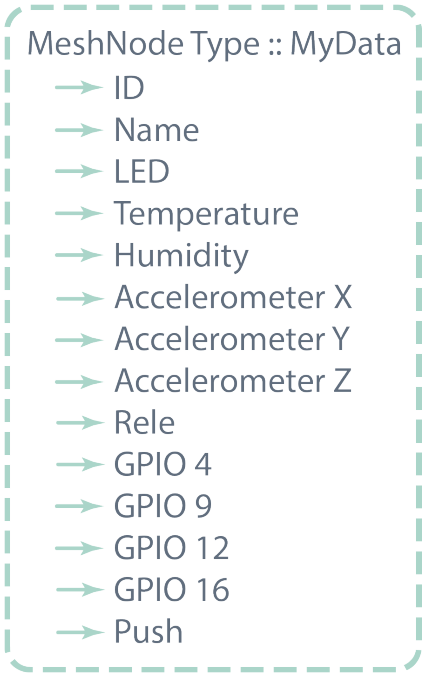


Imagen Nº64: Estructura de datos del nodo

Esta estructura es la base de funcionamiento del *firmware* ya que controla los estados de las salidas y toma los datos de las entradas generadas. Para poder efectuar esto se necesitan 2 tipos de funciones, un primer grupo encargado de tomar los valores de los sensores y establecer los estados de las salidas y un segundo grupo el cual tiene a cargo la comunicación con la **red mesh** para poder actualizar los valores de las salidas y enviar información de entrada.

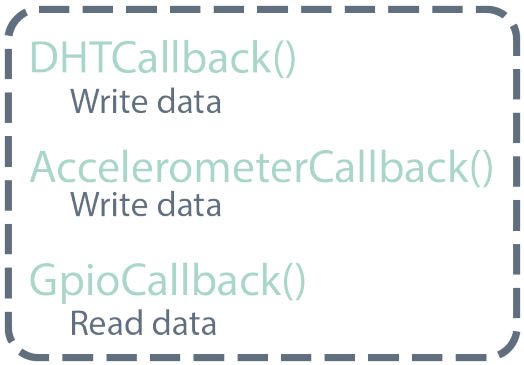


Imagen Nº65: Funciones de manejo de periféricos

Como se puede apreciar en la **Imagen Nº65** este grupo de funciones a su vez se divide en 3 funciones *callback*:

- DHTCallback(): Es la encargada de la lectura del sensor de temperatura y humedad. Es una función únicamente de escritura ya que solo toma datos del sensor y los almacena en el atributo de la estructura destinada a tal fin. No tiene ningún control sobre las salidas ni la comunicación.

- AccelerometerCallback(): Es la encargada de la lectura del sensor acelerómetro y giróscopo. Es una función únicamente de escritura ya que solo toma datos del sensor y los almacena en el atributo de la estructura destinada a tal fin. No tiene ningún control sobre las salidas ni la comunicación. En el caso de esta implementación no se hace uso de los valores de giro que aporta el sensor pero si pueden ser añadidos en un futuro prototipo.

- GPIOCallback(): Es la encargada de la lectura y escritura de las entradas y salidas digitales GPIO. Es una función de escritura y lectura ya que dependiendo de la configuración que se le de a cada uno de los GPIO puede escribir o leer dicho pin. En el caso de esta implementación la función esta configurada únicamente en su estado de lectura ya que lee los valores que debe tener cada una de las entradas/salidas en la estructura de datos y como consecuencia, ajusta el estado. No tiene ningún control sobre las salidas ni la comunicación.

Por último se encuentran aquellas funciones destinadas a la comunicación con la red mesh, las mismas no tienen acceso al control o lectura de los periféricos del microcontrolador pero si son las encargadas de leer, enviar, interpretar y recibir toda la información que llegue y salga del nodo.

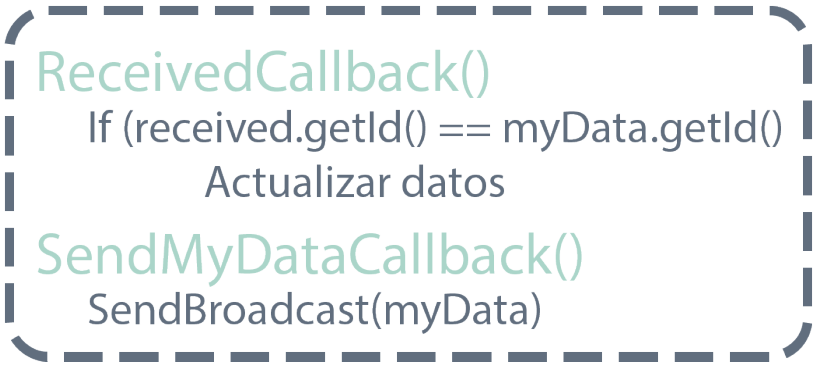


Imagen Nº66: Funciones de mesh

Se puede apreciar que dentro de este grupo existen dos funciones encargadas del proceso de comunicación con la red mesh:

- ReceivedCallback(): Esta función es invocada cada vez que un nuevo mensaje llega a la red entregando como respuesta el mensaje recibido y el ID del nodo que envió dicho

mensaje. Los mensajes que se mueven a través de la red contienen los mismos datos que la estructura MeshNode en el *firmware* de los nodos (**Imagen Nº65**) pero codificada. Es por eso que al llegar un nuevo mensaje se necesita descodificar al mismo para poder obtener la ID que se encuentra en ese mensaje, si este número coincide con el numero único del nodo que recibió el mensaje la información es de importancia y se procede con ella. De no ser así significa que el mensaje no era para ese nodo por lo que este es descartado y queda a la espera de uno nuevo. Suponiendo un mensaje exitoso se actualiza la información actual del nodo con la información recibida pero no en su totalidad dado que simplemente se necesitan los datos que tengan que ver con los periféricos de salida del dispositivo.

- SendDataCallback(): Esta función a diferencia de la anterior tiene un funcionamiento mucho más simple de entender, es la encargada de codificar los datos del nodo y enviarlos a la red mesh.

SOFTWARE HUB

La idea detrás del funcionamiento del hub es la de manejar toda la información que se encuentra en la red y darle una estructura y uso. A diferencia del *software* que tienen los nodos, el módulo hub no tiene que guardar la información de sus periféricos sino simplemente almacenar y controlar los datos de todos los nodos que estén en la red y, a su vez tener una comunicación con el módulo de thinger.

Es por eso que este dispositivo genera una lista de estructuras MeshNode en la cual cada uno de los integrantes de esta lista en un nodo en la red. Un ejemplo sería si en la red existen 4 nodos conectados, el hub tendrá una lista con 4 espacios donde se almacenará la información de nada uno de esos nodos.

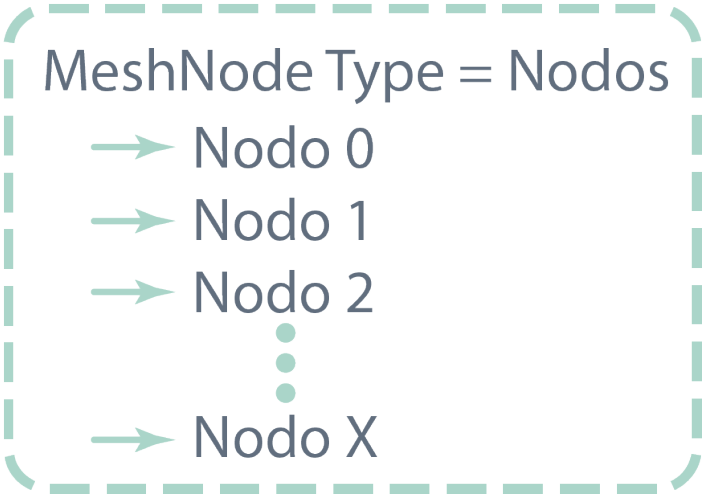


Imagen Nº67: Lista de nodos

Como se mencionó anteriormente y tal cual se puede ver en la **Imagen Nº67** este dispositivo es el encargado de concentrar toda la información de los nodos que se encuentran en la red en un único lugar. Para eso se van a necesitan 2 tipos de funciones, ambas encargadas de comunicaciones y de enviar y recibir información. El primero grupo de funciones estará a cargo de las comunicaciones con la **red mesh** y el segundo grupo lo hará con el dispositivo thinger.

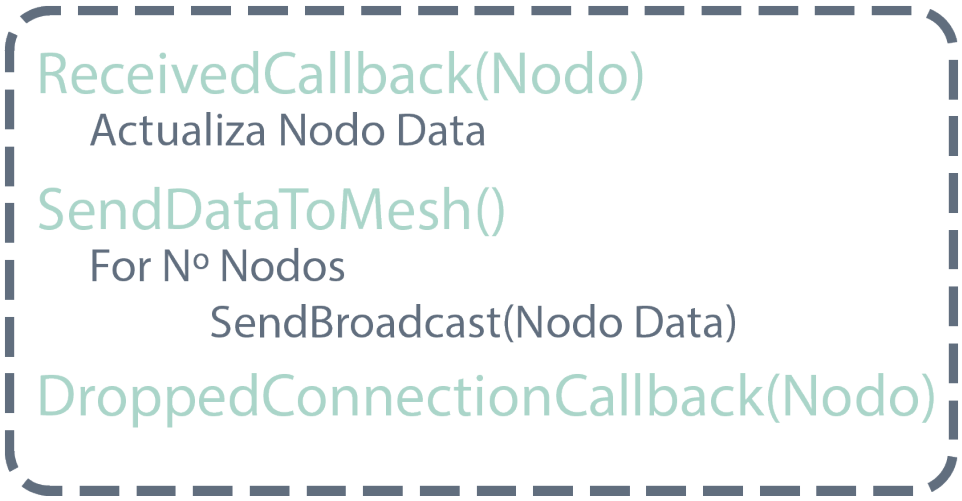


Imagen Nº68: Funciones de mesh

Como se puede ver, este conjunto está formado por dos funciones distintas que son las encargadas de las comunicaciones con la red mesh:

- ReceivedCallback(Nodo): Esta función es invocada cada vez que un nuevo mensaje llega a la red entregando como respuesta el mensaje recibido y el ID del nodo que envió dicho mensaje. Los mensajes que se mueven a través de la red contienen los mismos datos que la estructura MeshNode en el *firmware* de los nodos (**Imagen Nº64**) pero codificada. En este caso a diferencia de los nodos, el hub no necesita comprobar si la información es para el ya que toda la información que reciba es de importancia y proviene de algún nodo de la red. Al recibir un mensaje se lo decodifica y se de comprueba el nombre del nodo que envió el mensaje, con esto se pueden actualizar los datos en la lista de nodos.

- SendDataToMesh(): Al tener una lista de nodos se necesita poder recorrer la lista, una vez tenido cada uno de los elementos se los codifica y enviar a la red. El envío del mensaje no es de forma directa sino que se envía a toda la red, luego cada nodo tiene la capacidad de distinguir el correcto destinatario del mensaje.

- DroppedConnectionCallback(Nodo): Esta función tiene un único fin y es eliminar el nodo de la lista en el caso de que el mismo se desconecte de la red, reduciendo la lista en un elemento.

Por último se encuentran aquellas funciones que se encargan de la comunicación con el dispositivo thinger. Las mismas no tiene control sobre la comunicación con la **red mesh** por lo que su fin es exclusivo de la comunicación UART con dicho módulo.

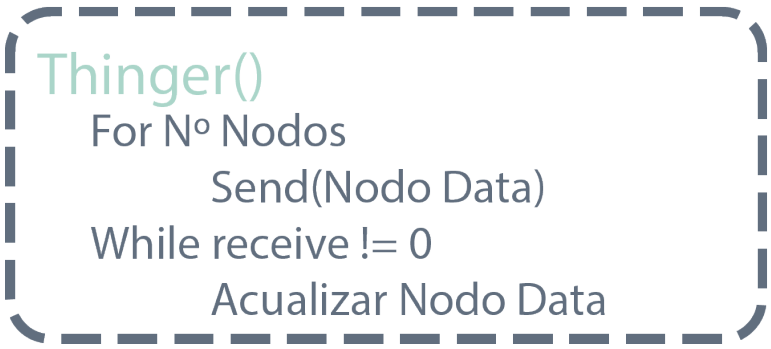


Imagen Nº69: Funciones de thinger

Como se puede ver en solamente se tiene una única función encargada de la comunicación, esto se debe a la búsqueda de un orden en el envío y recibo de datos y que no se generan sin un orden establecido ya que se puede perder información y aumentar los tiempos de actuación.

- Thinger(): Como se mencionó anteriormente esta es la función encargada de la comunicación por medio de la UART entre el dispositivo hub y el dispositivo thinger. La función comienza recorriendo la lista de nodos y a medida que se los va obteniendo, se los codifica y enviar al otro dispositivo. Una vez terminado se procede a enviar una señal de “ok” que simboliza que el dispositivo hub finalizo con el envío de datos por lo que el dispositivo thinger puede proceder a hacer el envío de sus datos. A continuación se queda a la espera de datos en el canal UART y a medida que se va recibiendo se actualizan los valores de la lista. En este caso los únicos datos de cada nodo que son necesarios son los que tengan que ver con la actuación de periféricos de salida.

SOFTWARE THINGER.IO

Este dispositivo es el único de los 3 que no tiene posible acceso a la **red mesh**, esto se debe a que es el encargado de tener una conexión a internet para conectarse a los servidores de **Thinger.io**.

Dado que todos los dispositivo cuentan con una única antena de comunicación WiFi es que no tiene la capacidad de estar conectados a la **red mesh** y a una red de internet al

mismo tiempo, es por eso que se optó por incorporar este módulo que sirve como puente entre la **red mesh** y los servidores alojados en la nube.

Su funcionamiento se basa en recibir la información desde el dispositivo hub a través de una comunicación UART y, al igual que el hub, crear una lista con todos los nodos que existan en la red. Usando las librerías brindadas por el fabricante **Thinger.io** es que se logra crear *APIs* que el servidor web interpreta y muestra en forma gráfica para que el usuario pueda controlar la red y ver la información de la misma.

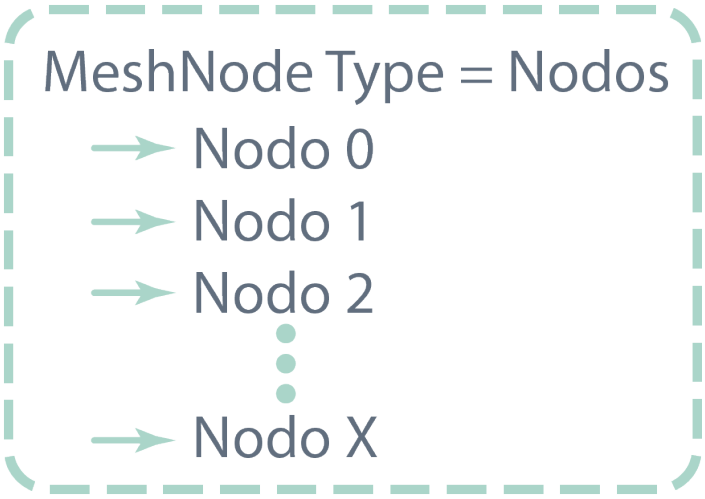


Imagen Nº70: Lista de nodos

Al igual que se vio en el dispositivo hub, en este se encuentra básicamente la misma lista de nodos con la misma información. En este caso también se necesitan dos sets de funciones que serán las encargadas de por un lado tener una comunicación estable con el hub, y por otro establecer una conexión exitosa con el servidor web y la creación de las APIs que se convertirán en el *front-end* de todo este sistema. Las APIs generadas permiten vuelven a este un sistema fácil de controlar y entender hasta para aquellos usuarios que no son adentrado en el mundo de la tecnología.

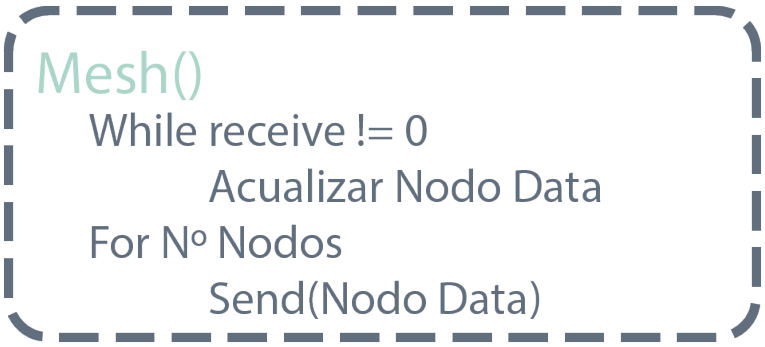


Imagen Nº71: Funciones de hub

El primer conjunto simplemente cuenta con una única función la cual se encarga, al igual que en el caso del hub, de estable una comunicación estable y ordenada por medio de la UART entre estos dos dispositivo.

- Mesh(): Que el nombre no deje engañar, esta función no se encarga de establecer una conexión a ninguna **red mesh**. El nombre esta dando debido a que esta función genera la comunicación con el dispositivo que tiene acceso a la **red mesh**. Si se recuerda lo visto en la **Imagen Nº71**, este caso es completamente lo opuesto. Aquí la función queda a la espera de recibir mensajes por medio de la UART, al momento de recibir irá almacenando todos estos datos en el nodo correspondiente teniendo actualizados constantemente la información pertinente a cada uno. Luego de esto se recibida la señal de "ok" para simbolizar que es momento nuestro momento de enviar datos. Se procede recorriendo la lista de nodos, una vez obtenido cada uno se lo codifica y envía, así con cada uno de los nodos en la lista. Una vez finalizado se envía la señal de "done" para decirle al dispositivo hub que se acabó en envío de datos y puede comenzar de nuevo.

Para finalizar se encuentran una de las partes más importantes de todo el sistema, las *APIs*. Este último conjunto no tiene acceso a la comunicación que se genera con el módulo hub sino que simplemente es la encargada de plasmar toda la información en una forma en la que el servidor la pueda interpretar correctamente.

Poniendo en contexto, una *API* o interfaz de programación de aplicaciones, conocida por sus siglas en ingles "*Application programming interface*" es un conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta librería para ser utilizado por otro *software* como una capa de abstracción.

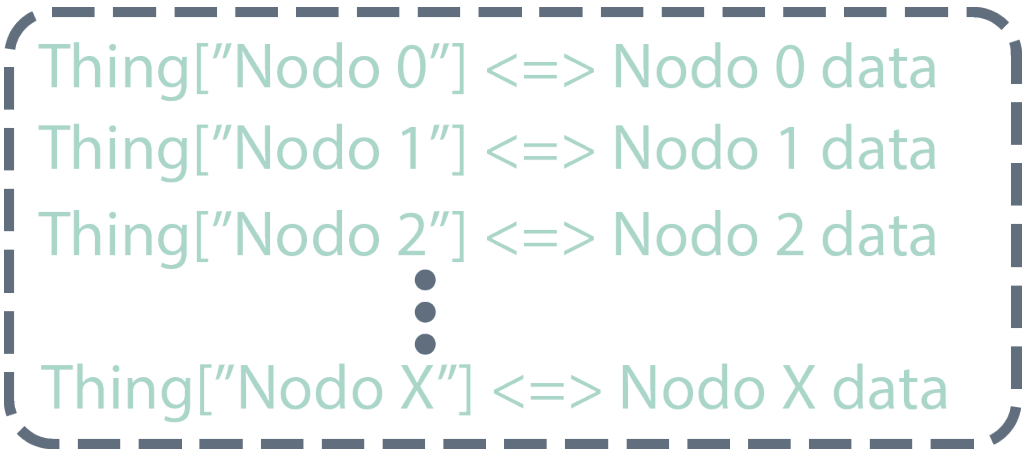


Imagen Nº72: Creación de APIs

La forma en la que se crean las *APIs* esta establecida por el fabricante en su apropiada documentación pero se basa en la creación de un objeto "*Thing*" el cual posee un nombre

(ejemplo “Nodo 0”). Luego de esto se debe especificar si será una *API* de toma de datos “<“, de escritura de datos “>” o de ambas “<=>”. La documentación permite la creación de otros tipos pero estos 3 son los más básicos y utilizados en este proyecto. Una vez dentro de la propia estructura se debe indicar cuales serán los valores que se mostraran como entrada y cuales como salida, todo esto se lo hace en formato **PSON**.

PSON no se diferencia en nada con respecto a **JSON**, la cual es un acrónimo de “*JavaScript Objerct Notation”.* Esta notación de objeto de **JavaScript** es un formato de texto sencillo para el intercambio de datos. Se trata de un subconjunto de la notación literal de objetos en **JavaScript**, aunque, debido a su amplia adopción como alterativa **XML**, se considera un formato independiente del lenguaje.

```
Thing["Node"] = [](pson& in, pson& out)
    → Out["ID"] = node.getId()
    → Out["Node number"] = node.getName()
    → Node.setLed((int)in["Led"])
    → Out["Temperature"] = node.getTemperature()
    → Out["Humidity"] = node.getHumidity()
    → Out["Acceleration in X"] = node.getAccelerationX()
    → Out["Acceleration in Y"] = node.getAccelerationY()
    → Out["Acceleration in Z"] = node.getAccelerationZ()
    → Node.setRele((bool)in["Rele"])
    → Node.setGPIO4((bool)in["GPIO4"])
    → Node.setGPIO9((bool)in["GPIO9"])
    → Node.setGPIO12((bool)in["GPIO12"])
    → Node.setGPIO16((bool)in["GPIO16"])
    → Out["Push"] = node.getPush()
```

Imagen Nº73: Estructura de API

En la **Imagen Nº73** se puede apreciar la estructura que tienen las *APIs* del sistema. Si se recuerda cómo estaba conformado cada una de las esculturas de datos de los nodos se puede ver que es prácticamente igual pero agregando los comandos que le dan vida a esta interfaz de programación. Nuevamente se tiene que mencionar que esto es simplemente una implementación por lo que pueden ser añadidos más datos dependiendo del gusto del usuario y los periféricos con los que cuente el nodo.

Como se mencionó a lo largo de este informe la plataforma usada en este trabajo final de grado es **Thinger.io**, la cual cuenta con una interfaz de usuario muy amigable y fácil de entender. A continuación se procederá a mostrar como el usuario interactúa con la red por medio de esta plataforma tanto en un navegador web como en un dispositivo Android.

Interfaz de dispositivos

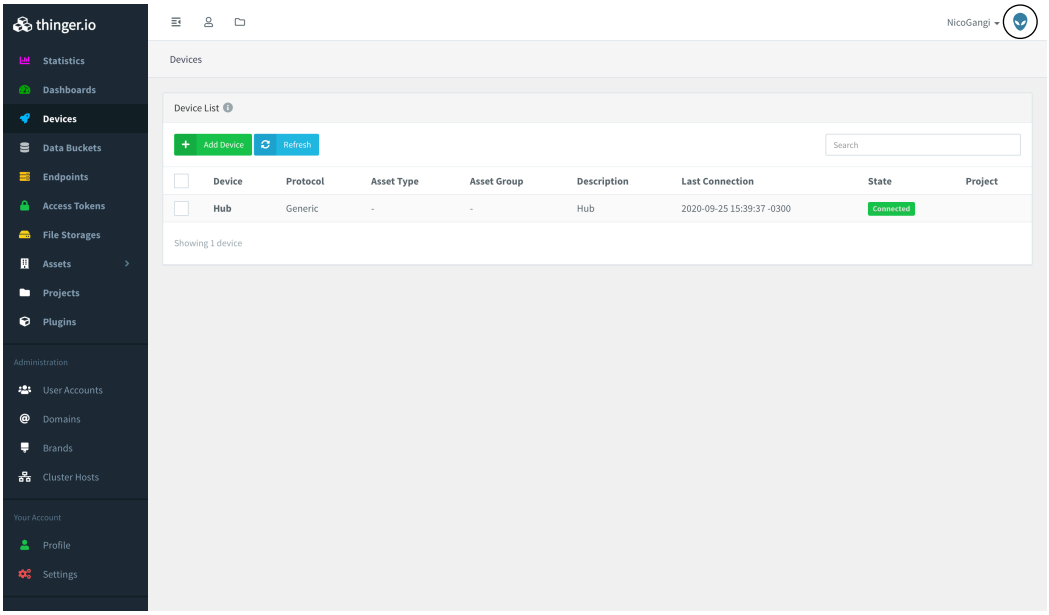


Imagen Nº74: Plataforma en navegador web

En la **Imagen Nº74** se puede observar la pantalla principal que tiene el usuario al momento de conectarse por medio de un navegado web. Se puede apreciar que la misma permite tener varias dispositivos, esto se debe a que es una plataforma de dispositivos **IoT** por lo que se puede simplemente agregar un nuevo dispositivo. En el caso de este proyecto toda nuestra red se representa por medio de un único ícono que centraliza todo y se lo nombro “hub” pero el nombre puede ser puesto por el usuario.

A la derecha del nombre se puede apreciar cierta información a cerca del dispositivo pero la más importante a recalcar es la descripción la cual puede ser también puesta por el usuario, fecha y hora de la última conexión y el estado el cual varia entre conectado y desconectado.

En el caso de no ser el propietario del dispositivo pero tener un *token* generado por el administrador del mismo, este aparecerá de igual manera y se mostrará la misma información durante el tiempo en el que dure el acceso. Hay que aclarar que los *token* de acceso se conceden por medio de un código QR generado por el administrador y el mismo puede establecer el acceso y el tiempo del mismo.

Interfaz de dispositivos

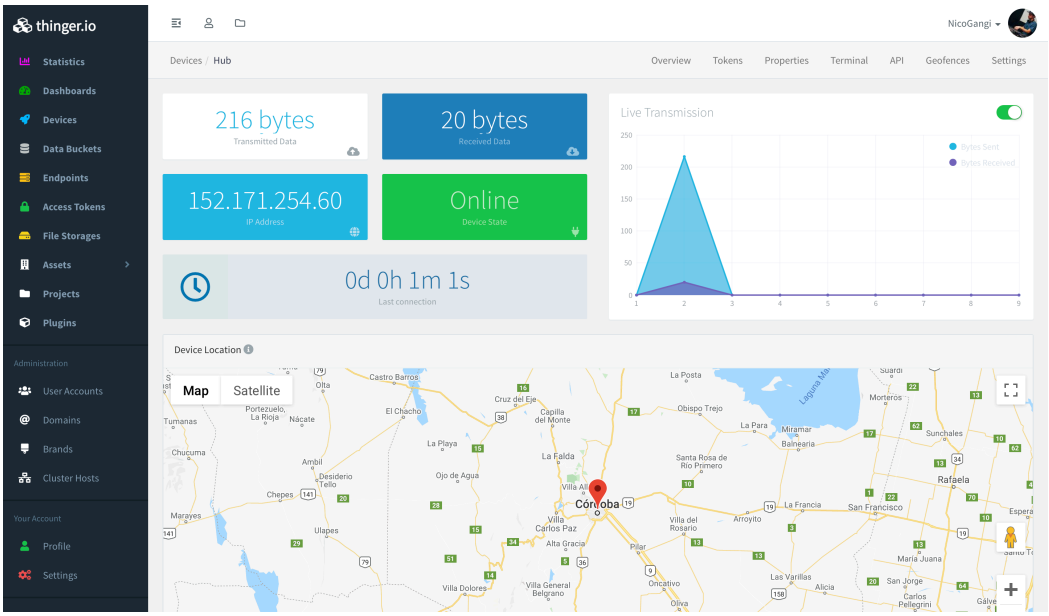


Imagen Nº75: Dispositivo en navegador web

Una vez ingresado dentro del dispositivo, en este caso “hub” se obtendrá una pantalla parecida a la de la **Imagen Nº75**. En ella se pueden notar 3 grandes grupos de información. En un primero lugar, en el margen superior izquierdo se muestra datos tales como cantidad de datos transmitidos [bytes], cantidad de datos recibidos [bytes], la dirección IP del módulo, el estado del dispositivo (conectado-desconectado) y por último el tiempo desde la ultima conexión.

Opuesto a esto, en el margen superior derecho se puede ver un gráfico en vivo de la transmisión en donde el eje de las ordenadas se representan la cantidad de bytes enviados y la cantidad de bytes recibidos; por otro lado en el eje de las abscisa se trafica el tiempo.

Por último en la parte de abajo de la pantalla se puede apreciar un mapa interactivo en el cual se puede ver la ubicación aproximada en donde se encuentra el dispositivo, no la ubicación del usuario.

Interfaz de APIs

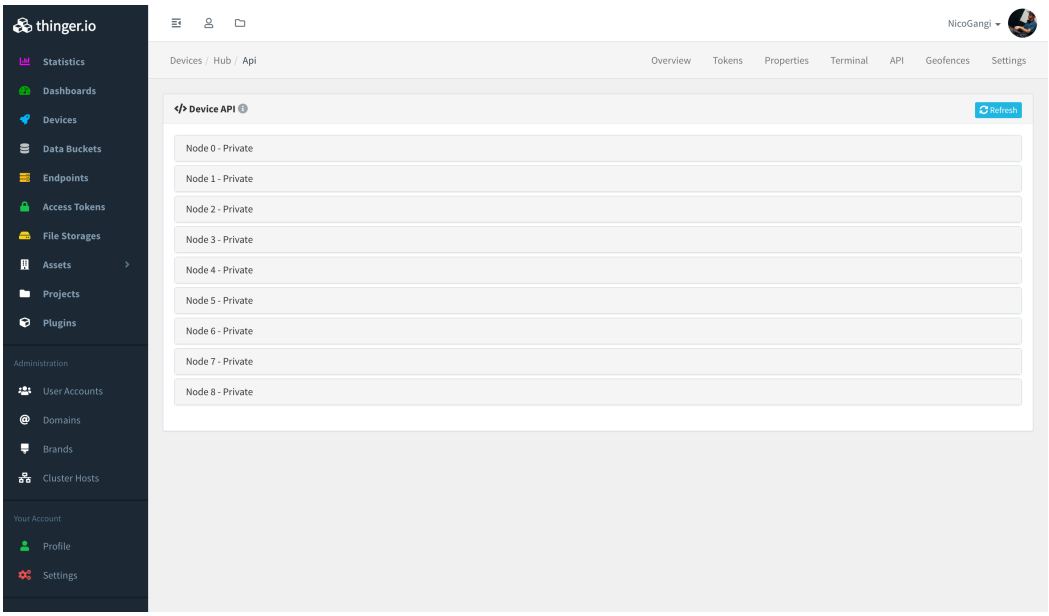


Imagen Nº76: APIs en navegador web

En la **Imagen Nº76** se puede apreciar cómo la plataforma muestra al usuario las *APIs*. En el caso de esta implementación se decidió por la creación de 8 instancias para representar los 8 nodos de la red. En un futuro prototipo se puede pensar en una creación de automática de interfaces de programación para que solo aparezcas los nodos conectados a la red.

Interfaz de nodo

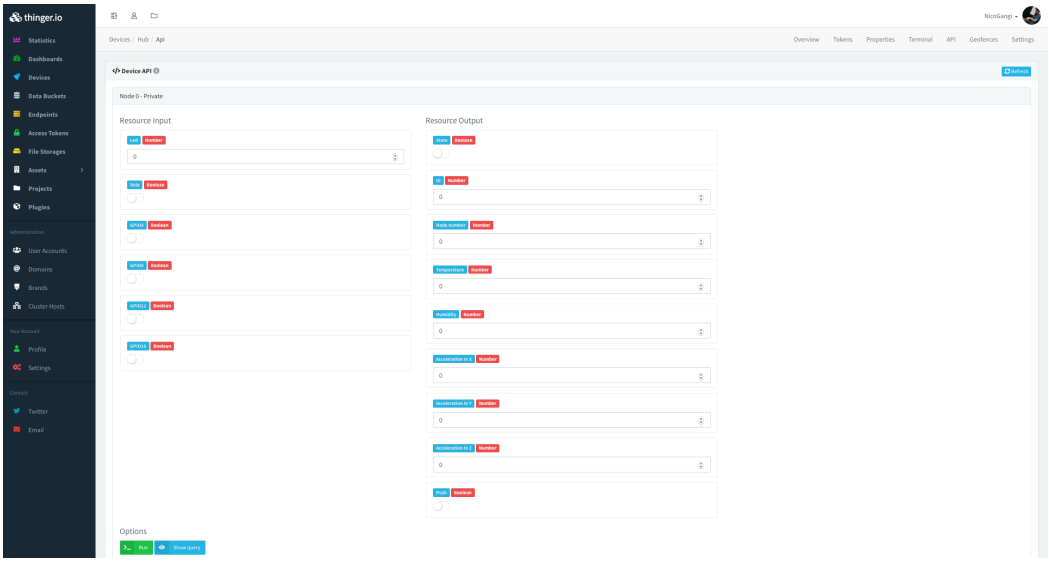


Imagen Nº77: Nodo en navegador web

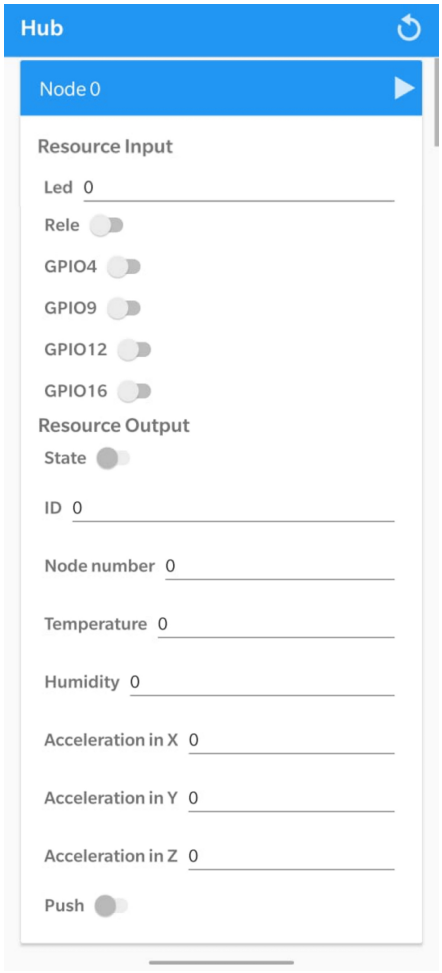


Imagen Nº78: Nodos en dispositivo Android

Como se mencionó con anterioridad, la plataforma **Thinger.io** también cuenta con una aplicación móvil para dispositivos Android que, a diferencia de la interfaz web, únicamente tiene una ventana y es la mostrada en la **Imagen Nº74**. Por contraparte, su equivalente en navegador web se puede observar en **Imagen Nº77**. Puede observarse que la versión web es un poco más atractiva a la vista debido a los colores y formas que posee pero, en su defensa la versión para dispositivos móviles tiene las mismas funcionalidades.

Un detalle a aclara es que el orden que la aplicación Android le da a los nodos no es en orden creciente o decreciente por lo que se tiene que buscar nodo por nodo hasta encontrar el que se quiere. En la versión web los nodos se encuentran por ordenados por nombre tal cual se indica dentro del *firmware* de thinger (**Imagen Nº72**).

En estas interfaz se puede apreciar en ambos casos cómo el fabricante divide las entradas con respecto de las salidas teniendo una visión rápida de lo que se necesite. En ella se pueden apreciar:

- Recursos de entrada:
    - LED
    - Rele
    - GPIO4
    - GPIO9
    - GPIO12
    - GPIO16
- Recursos de salida:
    - State
    - ID
    - Node number
    - Temperature
    - Humidity
    - Acceleration in X
    - Acceleration in Y
    - Acceleration in Z
    - Push

Nótese que el recurso de salida "*State*" sirve para simbolizar si el nodo en cuestión se encuentra conectado o no, siento verdadero en el caso de estar conectado a la red y falso en caso contrario.

Prestando atención se puede observar que también existe un botón llamado con las siglas "*Run*" (o el símbolo de un triangulo en la aplicación móvil). Este pulsador es el encargado de refrescar y enviar la información. Poniendo un ejemplo, se quiere refrescar la información de temperatura y humedad en cierto nodo, solo basta con apretar dicho botón y los últimos valores que se tengan aparecen en pantalla. Caso opuesto si se quiere cambiar el estado de una salida, se necesita cambiarlo en la interfaz gráfica y luego presionar el botón *Run* para enviar esta información hacia el módulo thinger y, posteriormente a la red.

Beneficios post implementación

La idea de implementar **redes mesh** para internet de las cosas surgió más que para resolver una problema como una opción o mejora a lo que se tiene hoy en día y para ocupar un lugar en el mercado que hoy en día no está del todo explotado. Pero el objetivo último de este trabajo final radica más que nada en la idea de juntar dos conceptos y volverlos un único producto, por un lado la potencia y versatilidad de las **redes mesh**, y por el otro el potencial crecimiento en el mercado de los dispositivos **IoT**.

Para caso domiciliario, tener la capacidad de equipar todo un hogar con dispositivos inteligentes y poder controlar la totalidad de las partes de una casa desde cualquier sitio sin importar al alcance del router WiFi es una de las ventajas de esta implementación. Permitiendo dar acceso a controlar y tomar datos de ciertos lugares que antes eran impensados dado a una limitación en el alcance de la red actual.

En cuanto al ámbito empresarial, esta implementación puede hacer ahorrar al empleador mucho dinero ya que permite hacer servicios de mantenimiento de forma remota sin la necesidad de un empleado destinado a eso o sin tener que enviar a un empleado al lugar poniéndolo en riesgo. Por otro lado tener un control del consumo eléctrico ayudará nuevamente al ahorro energético y económico de la empresa ya que es posible mantener luces y equipamientos apagados o prendidos dependiendo de la necesidad que se tenga en cada momento.

Por último en el caso rural, poder ser capaz de hacer un seguimiento constante de animales o cultivo con datos en tiempo real permite al propietario una actuación más selectiva. Ya sea un sistema de riego en el caso de un campo con cultivos o predecir comportamientos o ubicación de ciertos animales. Estos datos a su vez pueden ser sumados a un sistema de inteligencia artificial para poder así tener un conjunto de automatizaciones más profundo y profesional que permita al dueño una mejor explotación y cuidado de los recursos.

No hay que olvidar la variable económica la cual es un punto muy importante al momento de entrar a un mercado en donde los competidores buscan sacar nuevos productos cada día por un menor precio. La diferencia con este proyecto radica en que se busca llevar al mercado un producto innovador, versátil, pequeño, funcional y sobre todo adaptable a las condiciones del usuario.

Impacto social

Forbes es una revista especializada en el mundo de los negocios y las finanzas fundada en el año 1917 por B.C. Forbes, en Estados Unidos. También cuenta con una edición en España publicada por **SpainMedia**. El día 18 de junio del año 2018 **Forbes** hizo una publicación titulada "¿Cómo está ayudando la tecnología **IoT** al medio ambiente?", en ella hacen referencia a la importancia de este tipo de tecnología en la actualidad y como la misma aunque no se crea ayuda al cuidado del medio ambiente.

Afortunadamente a medida que pasan los años cada vez el ser humano esta tomando más conciencia sobre temas medioambientales y aunque parezca algo totalmente alejado de la tecnología **IoT**, la nueva revolución tecnológica, está ayudando al medio ambiente en más aspectos de los que se podría creer. Un buen ejemplo de ello es cómo ayuda a mantener la fauna salvaje o como ayuda a optimizar los recursos.

En el caso del entorno rural hay que destacar los dispositivos de internet de las cosas encargados de detectar incendios forestales. Los mismos cuentan con sensores que detectan la humedad, temperatura y gases de combustión que se encuentran en una zona especifica y que activan una alerta cuando la zona está en peligro permitiendo así actuar rápidamente y que las consecuencias sean nulas o mínimas.

También hay que hablar de cómo esta tecnología está ayudando a la fauna salvaje, por ejemplo ayudando a restablecer el control de especies en peligro de extinción que gracias a la tecnología **IoT**, y mediante unos collares con GPS, se puede monitorizar a los pocos integrantes que queden, conociendo así datos como su posición o actividad.

Por otro lado, estos módulos también ayudan a la fauna salvaje disminuyendo el daño causado por la casa furtiva. En algunas áreas de África donde la casa es una practica muy habitual, aunque ilegal, lo que se ha hecho es dotar a algunas especies de rastreadores GPS, cámara y monitores de frecuencia cardiaca, de manera que cuando se registran datos sobre un nivel alarmante en los animales, los investigadores puedan intervenir inmediatamente.

Sin embargo, el internet de las cosas también puede ayudar de forma indirecta al ecosistema dado que, en las fabricas del futuro, el medio ambiente jugará un papel sumamente importante. Y este tipo de dispositivos ayudarán a vigilar los niveles de calidad de aire, y del agua haciendo que estos establecimientos puedan saber si están sufriendo alguna fuga o si se están excediendo en la cantidad de emisiones contaminantes que expulsan. Esto hace que puedan intervenir rápidamente, reduciendo así el impacto y las

perdidas que puedan producir. Esto no es solo un beneficio para la fábrica en si, sino que también ayuda a disminuir la huella que está dejando en el medioambiente.

En un plano que puede tocarnos más cerca, no debemos olvidarnos de las casas y edificios inteligentes que, mediante sensores y otros dispositivos conectados están ayudando a reducir los consumos de recursos de manera muy considerable. Desde mejorar la eficiencia de los mismos hasta controlar cuando no se están utilizando. Esto supondría que en un futuro no muy lejano se puedan reducir los efectos del cambio climático hasta en un 16.5% aproximadamente según comenta la revista **Forbes**.

Estos son solo algunos ejemplos de cómo la tecnología puede ayudar a cuidar al medio ambiente, pero no todo está en sus manos. Y aunque es verdad que gracias a la tecnología el ser humano ha avanzado en el cuidado medioambiental, la pregunta que se debe hacer es ¿cómo estamos ayudando a proteger, preservar y restaurar el medio ambiente?

Estas son las razones consideradas a la hora de encarar el trabajo final y por las cuales se considera que no solo es una opción viable y con mucha entrada al mercado sino que al mismo tiempo que ayuda a la comodidad de los usuarios está cuidando al medio ambiente de una forma muy importante y es la forma en la que se debe encarar el diseño de las tecnologías modernas.

Conclusión

Anexos

If the positions recommended are not suitable, please make sure that the module is not covered by any metal shell. The antenna area of the module and the area 15 mm outside the antenna should be kept clean, (namely no copper, routing, components on it) as shown in Figure 1-14:

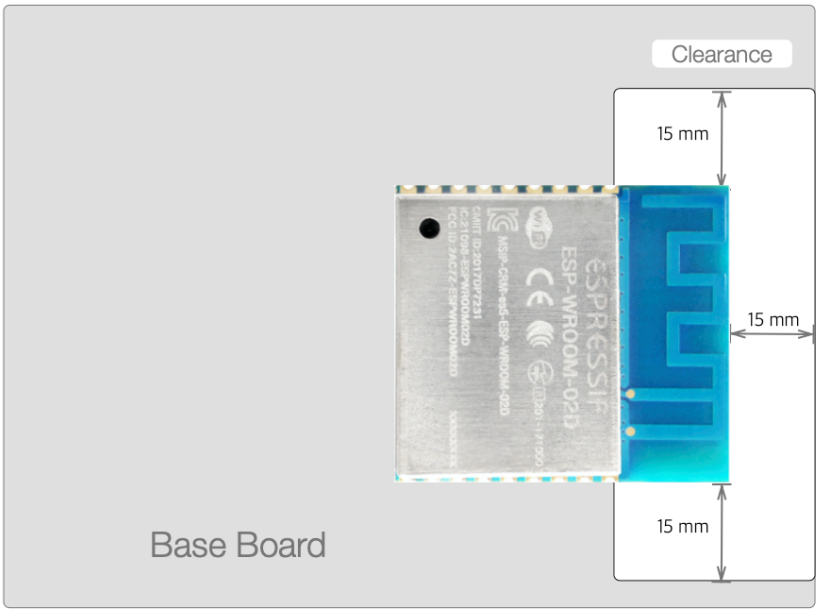


Figure 1-14. Keepout Zone for ESP8266 Module's Antenna on the Base Board

Wireless Mesh Networks in IoT Networks

Yu Liu
Department of Electronic and Electrical Engineering
University College London
London, United Kingdom
Yu.liu@ucl.ac.uk

Dr. Kin-Fai Tong
Department of Electronic and Electrical Engineering
University College London
London, United Kingdom
k.tong@ucl.ac.uk

***Abstract*—Internet of Things is one of the hottest topics in both industry and academia of the communication engineering world. On the other hand, wireless mesh networks, a network topology that has been discuss for decades that haven't been put into use in large scale, can make a difference when it comes to the network in the IoT world today. This paper is a brief introduction of how these technologies has the possibility to come together and how to integrate the mesh network into existing IoT networks to potentially make a difference in the new era.**

Keywords—IoT; Wireless Mesh Networks; sensors; wireless; network topology;

I. INTRODUCTION

With the significant growth of the semiconductor industry, creating small devices with powerful processing ability and network capabilities are no longer a dream for engineers. Currently, Internet of Things (IoT) has become one of the hottest topics in both industry and academia of wireless communication field. Today, most of the research of the IoT-enabled devices is mainly of the data collecting and processing units namely creating new sensors. However, the network that integrating the IoT devices to the Internet is usually left untouched by simply using existing computer network solutions such as WLAN or Bluetooth. These computer networks are not designed for low-powered devices such as remote sensors even these IoT devices are considered to be mini computers. The single point of failure nature of these Network makes the entire system extremely vulnerable when it comes to disasters or even difficult environment as the sensors may need to be deployed into some hardly reachable locations. Besides, the capacity of the central hub/router of the network can also limit the coverage of the service provided by IoT devices, and the range is also constrained by the same factors. As most of these remote IoT devices are small, and the devices are usually battery powered, so the power-hungry network options such as using cellular network or satellite are also not ideal for most of the remote scenarios in the IoT networks.

A wireless mesh network (WMN) is a communications network made up of radio nodes organized in a mesh topology instead of star topology used in most of the networks, according to Akyildiz, X. Wang in the book of Wireless Mesh Networks. [1] It is not a new concept at all, as it had emerged from the Multiple Ad Hoc Networks in the 70s from Packet Radio NETwork (PRNET) created by The Defense Advanced Research Projects Agency (DARPA) of the U.S. Department of Defense. [2]. Later in the 90s, many other civil solutions had also been proposed and created for different uses such as expanding the coverage of broadband services. The distributed network nature of the wireless mesh network with its simple configuration is ideal for be implement in the IoT networks to take advantage of its expanded range as well as keep the hardware design minimal using smaller network module. Such networks also are more robust in the harsh environment as the network are distributed with no single central point of failure. In this paper, the authors will discuss all of these features of WMNs in detail and why these features make WMNs ideal for the IoT networks over the traditional star networks as well as discussing the way of integrating the WMN into the existing IoT networks or design the IoT network with new feature from the beginning.

II. WIRELESS MESH NETWORKS

A. *Wireless Mesh Networks Introduction*

The main difference between WMNs and star networks WMNs are wireless networks, which have the ability of dynamically self-organizing and self-configuration, and with mesh connectivity automatically establishing among nodes in the network while the conventional star network has a star topology which means all the terminal nodes are connected to a single central point which connects to the upper level of the network. The Fig. 1illustrated the topology of two networks.

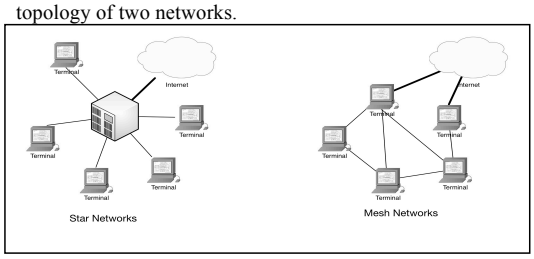


Fig. 1 Star Networks and Mesh networks

Currently, WMNs are adapted in several places, majorly in three different forms as follows:

1) Infrastructure/Backbone WMNs:

As shown in fig.2 this type of WMN includes mesh routers that form an infrastructure for clients that connect to them. The devices in the mesh router coverage areas still form a star network while the

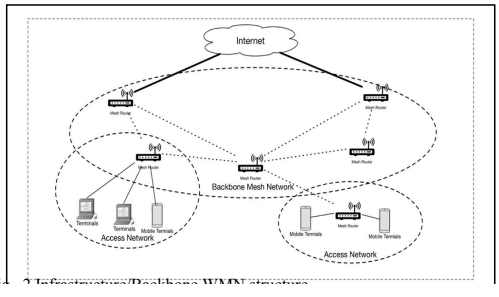


Fig. 2 Infrastructure/Backbone WMN structure

mesh routers form a mesh of self-configuring, self-healing links among themselves. With gateway functionality, mesh routers can be connected to the Internet. Infrastructure/Backbone WMNs are the most commonly used WMN as its simple and easy to integrate with the existing devices as only the routers need to be fitted to the mesh networks.

2) Client WMNs:
Client meshing provides peer-to-peer networks among client devices like a big ad-hoc network. In this type of architecture, client nodes constitute the actual network to perform routing and configuration functionalities as well as providing end-user applications to customers. Hence, a mesh router is not required for this type of network. This type of network is usually not accessible to the Internet.

3) Hybrid WMNs:

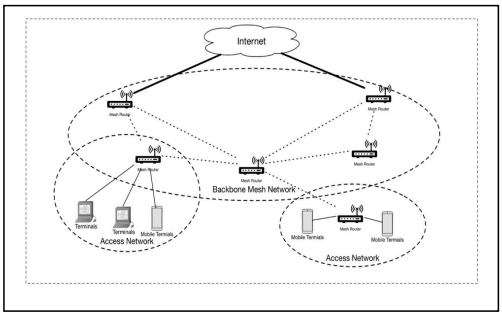


Fig. 3 Hybrid WMN structure

As shown in fig. 3 the hybrid mesh network architecture is the combination of infrastructure and client meshing as shown in the figure below. Mesh clients can access the network through mesh routers as well as directly meshing with other mesh clients. While the infrastructure provides connectivity to other networks such as the Internet, the routing capabilities of clients provide improved connectivity and coverage inside the WMN.

Only the infrastructure WMN and the hybrid WMN are suited for the network according to the requirement of different scenarios. Therefore, the discussion in the later part of this paper is mainly focus on these two kinds of WMNs.

B. Advantage of Mesh Network in IoT Networks

WMNs can bring many advantages to the IoT networks, and the most prominent one is the versatility of the network. When using the infrastructure WMN structure, adding a new router requires only simply putting the new device straightly into the field within the range of the existing network. The capacity and range of the network expand without introducing more cables and connections. For the hybrid network, this process is even simpler. Placing the new IoT devices in the field where the old mesh network is covered, the new sensor just works. It is because of the auto-configuration feature of the network will expand the network automatically. Hence, the network structure would be simpler, and the price of covering a larger area, especially in the rural area without reliable network everywhere could be much lower. Power consumption can also be significantly reduced when connecting remote IoT sensors or other devices to the network. The devices can only connect to closest mesh device instead of a distant central network hub, in the Hybrid case, it can even be a neighbouring node. This connects may make a chain of IoT devices, reducing the cost of power of the central hub to cover the most distant device.

Apart from the scaling and cost advantages when setting up new networks, WMNs also provide a more robust network for all kinds of applications in the IoT-world when it comes to unlikely events such as nature disasters. Even when some part of the link is destroyed and several devices are out of the connection. The rest of the network is still self-connected and self-configured with a new network and works the same. So long as one of the new network component connected to the internet, the entire sub-network will stay connected, only losing the connection from the node that is disconnected, whereas, in the conventional networks, this means the coverage of the entire area is lost, cutting off the IoT service totally.

C. Disadvantage of Mesh Networks in IoT network

Certainly, the WMN has some disadvantages. The unconventional network structure requires new network protocol support and the protocol should be compatible with the existing network as the IoT devices will eventually connect to the Internet. Besides, the repair of the network when a larger scale black-out is happening could be harder, as the cut-off the connection can hardly be detected when a larger network is disconnected. However, this can be solved by introducing error-checking report in the network packets, but at an expense of slower network. Lastly, the delay and the scalability of the network also limited by the nature of the mesh network. As proven by Belding-Royer EM et. al [3] in the famous paper, the delay and the data rate of the network also need to be limited. This may not be a problem in the IoT-devices, but needs to be considered when it comes to selecting the appropriate tool for the work required.

III. Integration of Mesh Networks in IoT Networks

This section is to show how to have mesh network enabled IoT network by integrating new network components into the existing networks using both infrastructure and hybrid network structures. Designing an IoT with mesh network in mind for the new development will also be introduced in the third part of this section to show the way to maximize the power of the mesh networks.

A. Infarstructure/Backbone Mesh networks

This is the simplest way of integrating WMN capabilities into the IoT networks. Illustrated in the fig. 2, the infrastructure/backbone WMN network works at the router level. The IoT networks using the traditional star network can easily migrate to WMN ready by simply switching the existing

routers/base stations on the edge of the network with mesh routers. This switch can introduce interconnectivity between the routers themselves instead of using an uplink which depends on the individual Internet connection from each router. After such migration, adding new routers into the current range of the network will simply scale up the network. However, as described in the disadvantage section, the network will suffer more delay and eventually, the network will be unusable when the hop number reach the magic number of six. [3]

This integration of the mesh networking into the existing IoT networks has already increased the scalability of the network without introducing significant investment.

B. Hybrid Mesh Networks

To integrate hybrid mesh network structure into the existing networks is more complicated but more rewarding. Instead of switching the routers/base stations, to take advantage of the versatility of the hybrid WMN, the network module of the individual IoT devices must be changed as well. The functionality of interconnectivity must be added to the modules. This can be done by reprogramming the module's controller. Some additional storage space is also preferable as the modules does not only store its own data, but the transferred data from the other modules is also required before the data gets forwarded. Broadcasting capabilities and timed network observation function are also essential to support the self-configuration feature of the WMN as every node is able to connect to the optimized node nearby when new node is introduced in the network. This can be done by adjust the RF module more promptly looking for available nodes when not connected. Integrating such capability will introduce more power consumption in each node, and this may cause the downgrade of service of certain IoT devices.

C. Designing the Meshed IoT Network from Ground Up

Apparently, the best way of integrating WMN into the IoT network is from very beginning. This means when designing the IoT devices, the integration with WMNs are considered. This includes two aspects, the end nodes and the network devices. Firstly, the end nodes, which are the IoT devices themselves. To design a WMN ready end node, the limitations of WMNs, especially Hybrid WMN, since the device itself will involve in forming the network, needs to be considered. The data package the device transmits each time shouldn't be too big as the speed limitation and the unpredictable of the WMN nature. Besides, the synchronize timing is also crucial to minimize collisions. Additionally, since most of the wireless IoT devices are battery powered, the dynamic network to provide mesh routing are more power hungry than conventional network modules, the optimization of the microcontroller is very important. The balance between the network and data processing is the key when designing a successful meshed IoT end node. Secondly, the network devices, since the network should be a hybrid network, which means each incoming connection to the mesh routes may not be only one devices. The MAC protocol is very important here to control the access of network both in bound and out bound. At the same time, it also needs to be compatible with the Internet as its the border device between the Internet and the meshed IoT network. When both network as well as end notes are designed with such WMN consideration in mind, the mesh ready IoT devices are just around the corner.

IV. Conclusion

As described in this paper, WMNs have all these great features using for the communication in the IoT networks. It is still under-developed as the industry advancing today. With the much more powerful MCU and processors today, the dynamic network topology can be achieved even in the tiny IoT devices. The mesh network topology has its unique advantage and disadvantage in the world of the IoT networks that can leverage the scale, distributed nature and low require of data-rate of the IoT devices. The advantage certainly outweighs the disadvantages of using the WMNs in such environment. Newer hybrid WMNs can be a solid choice when it comes to design the structure of the network, especially in the remote areas with its the robustness and scalability. This paper is simply discussing the possibility and the basic way of integrating such under-utilized network topology into the current and future IoT networks in the background of the advanced technology we have today. WMNs will certainly make a difference in the industry once being deployed on large scale in the IoT world and make the IoT more accessible to a wider audience.

References

[1] X. W. Ian Akyildiz, Wireless Mesh Networks, London: John Wiley & Sons, 2009.

[2] J. Okin, The Internet Revolution: The Not-for-Dummies Guide to the History, Technology, and Use of the Internet, 1 ed., Winter Harbor, ME: Ironbound Press, 2005.

[3] P. M. M.-S. ,. a. L. E. M. Elizabeth M. Royer, “An Analysis of the Optimum Node Density for Ad hoc Mobile Networks,” 2001

Glosario

Redes mesh: Una **Mesh Network** (Red de malla) o **Meshnet** es una topología de red local en la que los nodos de la estructura (es decir, puentes, *Switches* y otros dispositivos de infraestructura) se conectan de forma directa, dinámica y no jerárquica a tantos otros nodos como sea posible y cooperan entre si para encaminar eficazmente los datos desde y hacia los clientes.

Internet de las cosas: El **internet de las cosas** es un concepto que se refiere a la interconexión digital de objetos cotidianos con internet, en definitiva, es la conexión de internet más con objetos que con personas.

Thinger.io: **Thinger.io** es una plataforma de **IoT** de código abierto basada en la nube, desarrollada por Internet of **Thinger SL**, una empresa española cuyo objetivo es proporcionar tecnología IoT eficiente, consistente y fácil de usar.

Hub: Módulo central de la red encargado de recolectar toda la información de los nodos y enviarla al módulo encargado de la conexión con internet.

Nodos: Módulos conectados a la red encargados de la recolección de información para su posterior envío y accionamiento de periféricos.

Thinger: Módulo encargado de la comunicación con el servidor web y al cual le envía la información proveniente del hub y viceversa.

Bibliografía

https://internetofthingsagenda.techtarget.com/definition/IoT-device

https://www.softwaretestinghelp.com/iot-devices/

https://www.mediapost.com/publications/article/302663/north-american-consumers-to-have-13-connected-devi.html

https://robots.net/tech-reviews/top-iot-devices/

https://www.pcmag.com/picks/the-best-wi-fi-mesh-network-systems

https://www.t3.com/features/best-mesh-network

https://meshprj.com/en/

https://forbes.es/empresas/44043/como-esta-ayudando-la-tecnologia-iot-al-medio-ambiente/

https://www.esp8266.com/wiki/doku.php?id=esp8266-module-family

https://github.com/gmag11/painlessMesh

https://gitlab.com/painlessMesh/painlessMesh/-/wikis/bridge-between-mesh-and-another-network

https://boosthigh.com/mesh-network-for-iot-devices/

https://www.instructables.com/id/ESP-12F-ESP8266-Module-Minimal-Breadboard-for-Flas/

http://www.sinaptec.alomar.com.ar/2017/10/tutorial-23-esp8266-obtener-inclinacion.html

https://docs.thinger.io/quick-sart/coding/coding-guide#input-output-resources

https://docs.thinger.io

https://gitlab.com/painlessMesh/painlessMesh/-/wikis/home